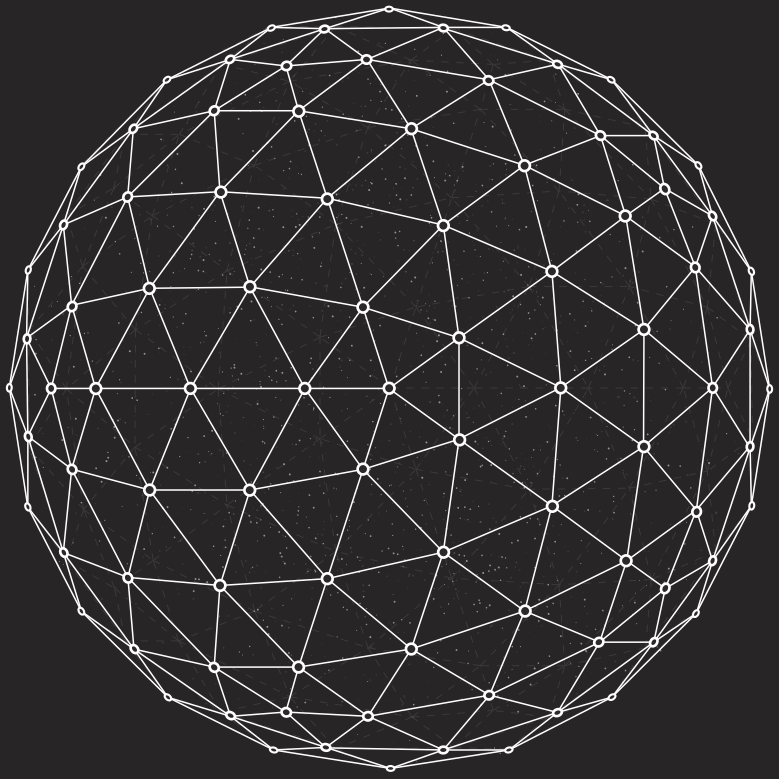


In memory of Daniel Ayre

DOWNLOAD THE ZINE FILES

You can use this address to download both the digital copy of NODE Vol 01, and all the project files:

dat://d5f52282d8277c323abcd838e7b1e62113af6dfa70f3c7316ec004911881ec41

CONTENTS

Editor: Mike Dank. Contributors: NODE, Sam Patterson, John Light, Burak Nehbit, Eric Schallock & Esteban Ordano

EDITOR'S LETTER

In 1968, a 29-year-old Stewart Brand and his wife embarked on a road trip in his 1963 Dodge pickup truck (dubbed the "Whole Earth Truck Store") to set up educational fairs at communes across America. Before the trip, Brand and his colleagues published the first issue of the *Whole Earth Catalog* (1968), subtitled "access to tools," to sell along the way. At each stop, Brand and his wife Lois created a mobile micro-education service, selling the catalog of tools, maps, clothing, computers, books, and journals, while offering advice to the growing, new communities. By the end of the trip, Brand's *Whole Earth Catalog* became a best-selling tool all in itself, and nurtured a growing counterculture movement with its curious, do-it-yourself attitude.

80% of the tools in the first *Whole Earth Catalog* were books. Written word--the transfer of knowledge from one person to another--is perhaps the most valuable tool we have as a species. These days, web content like video tutorials or podcasts has become a prominent way to dispense information, but the fundamentals aren't all that different from the texts of a bygone age.

Zines, small-circulation magazines often produced independently by a small group of people, have been around for decades. Going back to the 1930s and 1940s, many zines became staples within the science fiction community as more popular "pulp" sci-fi magazines would often reject more precarious stories that didn't align with their norm. Perhaps the biggest boom, or most radical mutation for zines, came in the 1970s by way of the punk subculture. The punk movement was underground, and the advent of home-publishing resources melded with the DIY spirit of punks around the world. Thousands of these punk zines lived and died throughout the '70s. Many were vulgar and profane, but filled with vital news on the burgeoning punk scene and other information that would build up punk rock culture. These zines were sold, or more often traded, around the world as a way to disseminate information to an otherwise unknown or isolated community. Punks didn't wait for the world to catch up--they didn't really want it to.

The Whole Earth Catalog
Photo by Akos Kokai (Creative Commons 2.0)

They took matters in their own hands, and created something they could control without outside influence or corruption.

The 1980s saw zines continue to flourish, expanding to the cyberpunks, grungers, and cynical Generation X slackers who hijacked their office photocopiers. Zines became not only a way to share your niche, but also escape from the monotony of your McJob. The real echo-boom in zine culture came with the Riot Grrrl movement, stemming from the marriage of DIY punk culture and American third-wave feminism in the early '90s. But, by the end of the decade, most prolific zines were either selling out and going corporate, or fading into nothingness as independent publishing companies went bankrupt. Almost overnight, a whole industry producing indie content disappeared, but that isn't the end of the story.

The advent of bulletin board systems (BBSes) in the 1980s led to an explosion of contemporary techno-culture. Counterculture pioneers and engineering wizards raised a new high frontier of interconnected computing. Computer users could dial into centralized machines and leave messages or share files for others to consume. It wasn't long before people started publishing eZines, electronic zines, for this new ethereal medium. By the time physical zines were on their way out in the '90s, the World Wide Web was hitting hard, spawning a fresh movement of independent content publishers that could inexpensively publish in cyberspace and have their work accessible to anyone with a modem. Content publishers and web developers danced in stunning coevolution--one group would feed off of and influence the other in a never-ending relationship we still see echoes of today. As time went on, we saw the birth of blogs, especially on free platforms, that gave way to the idea of **everyone** being a content creator. These days, the reverberations of zine culture are found in a lot of content we see and regularly consume. Podcasters, video-bloggers, indie game creators, lo-fi musicians, and others maybe have very well been zinesters back in the '80s and '90s. The spirit never died, it just adapted.

I've had the pleasure of working on NODE for a few years now, and I'm always excited to see how it evolves and grows over time. It took me too long to realize something about it that I

deem fundamental: NODE isn't a website or a video series or a person. NODE is an idea. NODE is a movement. NODE is a way to understand and reshape our world.

At its core, NODE works to gather and distill information to educate the curious. You're here because you're curious. You have a certain itch that just can't be scratched no matter how hard you try. NODE always pushes the envelope for how we can use technology to our advantage in this highly technical era. How can we make things more distributed? How can we put power back into the hands of the people? Every day, we see strides taken by corporations and government entities, but often at the cost of our personal liberty, privacy, or security. Hackers, tinkerers, developers, and makers just like you are building things on distributed technologies or shaking things up with movements like biohacking, 3d printing, and solarpunk, primed to disrupt the status quo and change our path for the better.

You're reading a zine right now. It's paper, and it's cold in your hands, and it's not as modern as you might be used to, but it continues in the footsteps of the DIY punk spirit and the Whole-Earthian standards of counterculture enlightenment. It's a perfect fusion of the eclectic idealism and self-sufficient attitude that shows what can be done when you have tools and the knowledge to use them. This zine holds firmly the idea of information sharing that NODE is built upon. It doesn't matter if you can make flashy videos or produce a website, all it takes is one person with an idea who can convey their thoughts to another. As the surveilled-web continues to grow along with our penchant to communicate digitally, print may become the only haven we have left to get information freely out of silos and into the hands of the motivated, standing the test of time.

If you're out there somewhere, we will be too. Knowledge is power, and we want to spread it.

I hope you enjoy this first issue of the annual NODE zine.

Stay hungry. Stay foolish.

Written by Mike Dank

INTRODUCING THE NODE MINI SERVER

The aim of the NODE Mini Server project is to create an inexpensive, easy to make hardware node which allows us to start building out the physical infrastructure for the P2P web. Here, we begin to replace remote servers with nodes that the users themselves own and operate.

Inside the NODE Mini Server is a Raspberry Pi 3B+, with future options of upgrading to the more powerful Asus Tinker Board. This choice allows you to run various node applications at the same time, depending on your needs. Bitcoin, Lightning and Ethereum nodes, payment servers, IPFS and dat servers, self hosting servers, and a bunch more applications are ideal for this platform. The internal 2.5-inch harddrive that's attached to the device should provide more than enough storage space.

You could easily put something similar together without modifying an SBC, but it would be a maze of wires, and much bigger.

This device measures in at 152x88x25mm, and can basically be plug and play.

The point of doing it this way is foremost to make a device that uses cheap, readily available parts, with many software options, but also so that anyone can build one. The hope is that other people begin to iterate and sell these so it becomes a standard and we can get them out to as many users as possible.

PARTS

- 3D Printed case halves (including struts which connect them)
- 3D Printed HDD/SSD frame
- Bottom PCB (1.6mm thick)
- Top PCB (1.6mm thick, optionally Aluminium-based)
- Micro SD Card Adapter PCB (0.8mm thick)
- SATA Adapter PCB (1.6mm thick)
- 8x M2.5 x 12mm Machine Screws
- 6x M2.5 x 8mm Machine Screws
- 6x M2.5 Hex Nuts
- 4x M3 x 6mm Machine Screws
- Rasbperry Pi 3B+

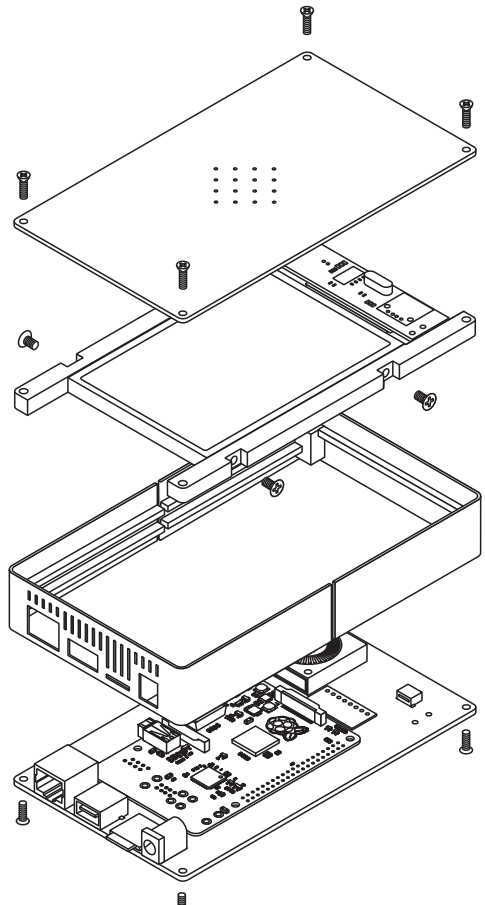
- 50mm 4pin 1mm Pitch FPC Cable (Contacts on same side)
- 2x 4pin 1mm Pitch FPC Connector (84981-4)
- 2.5-inch Hard Disk Drive or SSD
- Standard USB-A Port
- Standard SMD Micro SD Slot (Push/Push)
- 5.5mm x 2.1mm Right Angle DC Jack (694106301002)
- SMD RJ45 Jack (43743-8101)
- S8050 Transistor
- 30x10x10mm 5v Blower Fan (3010B)
- USB SATA Adapter (W25P1)

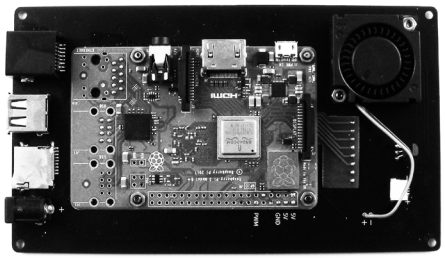
DESIGN

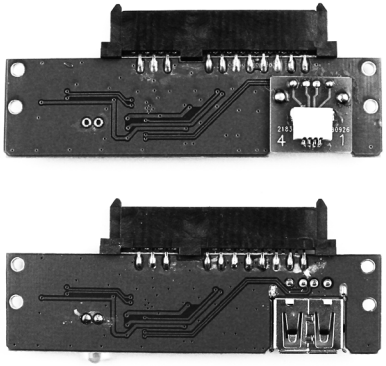
The case design consists of two PCBs sandwiched between a 3D-printed frame. This makes for a surprisingly sturdy and easy to assemble device.

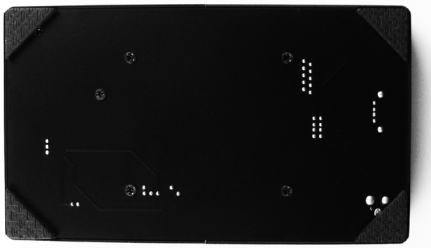
It's pretty minimalist, with just the screws and air holes drilled into the top case PCB.

There are no other defining features, other than the rear of the device, which has some more air vents as well as the ethernet, USB, power, and micro SD card ports.









Underneath, you can see the underside of the bottom PCB, and this design choice of exposing this board means the whole finished thing is much slimmer.

MODIFICATIONS

The internals comprise of two adapter boards, which basically connect the Pi to the 2.5-inch Hard Disk (or SSD).

Basically all of the modifications require something to desolder components. I used an electric vacuum pump, which makes life extremely easy, but if you only have a spring-loaded desolder pump and desolder wick that would work too.

Firstly on the Pi, we desolder the front three ports (ethernet, and 2x USB), and all the GPIO pins. You then plug in the micro SD adapter, screw the pi onto the board, and solder it all together with the corresponding holes underneath on the adapter board.

You then need to remove the USB port from the SATA adapter, and do the same procedure, replacing it with the mini PCB. This basically lets us connect the two halves together with the little flex cable.

Then, you simply solder all the other components onto the boards, connect the flex to the harddisk and Pi boards, and assemble and screw everything together.

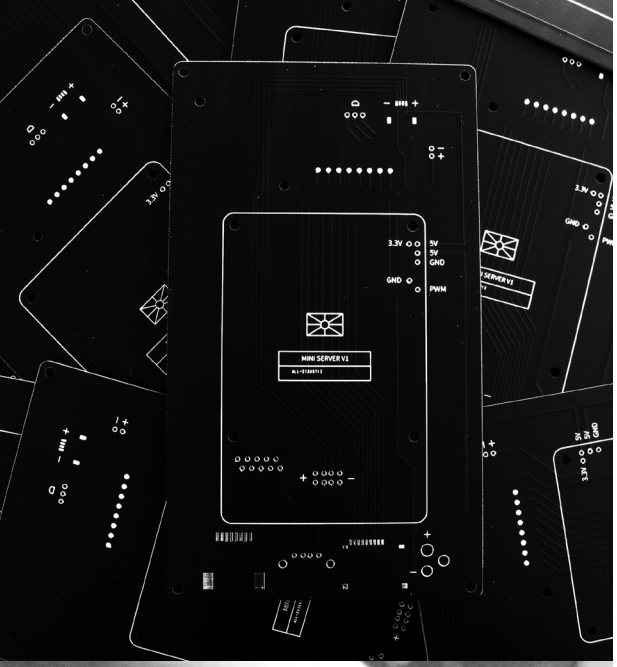
TEMPERATURE CONTROL

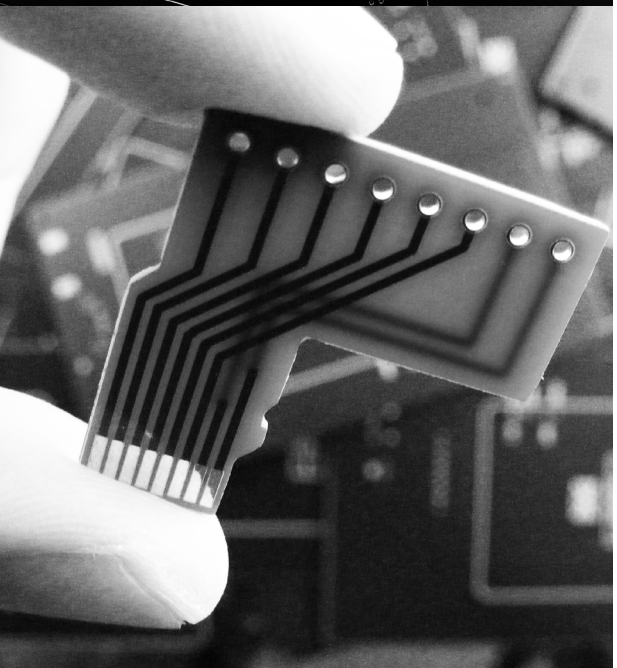
Little SBCs are often known for their heat generation, and the Pi 3B+ is no exception, so I've done a few things to keep it under control.

Firstly, the top PCB used on the case is aluminium-based, meaning under normal loads it stays pretty cool, and acts as a heatsink.

The micro SD card on the Pi has also been moved off the board itself, away from the heat, hopefully mitigating any of the temp-related problems they can suffer from.

There's also a temperature-controlled blower fan you can set up in Raspbian, which





automatically turns on when the Pi's CPU temp reaches a certain level, turning off again when it's cooled down.

The circuit design is the same used in a guide from HackerNoon [1], so follow that to set it up. Also, I found a better script than the one they used [2], so check that out too.

CONCLUSION

Hopefully this was worth the wait. I think we're almost at the point of having a truly great plug and play mini server, and I hope this nudges things closer. Once a faster Pi 4 comes out, it'll be perfect. If you try to make one let me know how it goes.

SOURCES

[1] `https://n-o-d-e.net/url/1/`
[2] `https://n-o-d-e.net/url/2/`

Written by NODE

DAT PROJECT 101: DECENTRALIZED DATA STORAGE

Dat is an open-source data distribution tool for easily duplicating and version-controlling different data sets. First released in 2013, Dat combines aspects of Git, BitTorrent, and cloud-storage sites like Dropbox to create an easy-to-use method for reliably sharing files without needing additional infrastructure.

HOW IT WORKS

Three important design features of Dat are in-place-archiving, file versioning, and the use of a distributed network.

Dat is most commonly used by installing the official Dat application and invoking it with commands through your terminal. There is also a GUI version available (pictured, right) that you can download from `https://datproject.org`

After installing Dat, let's see how a user can interact with it.

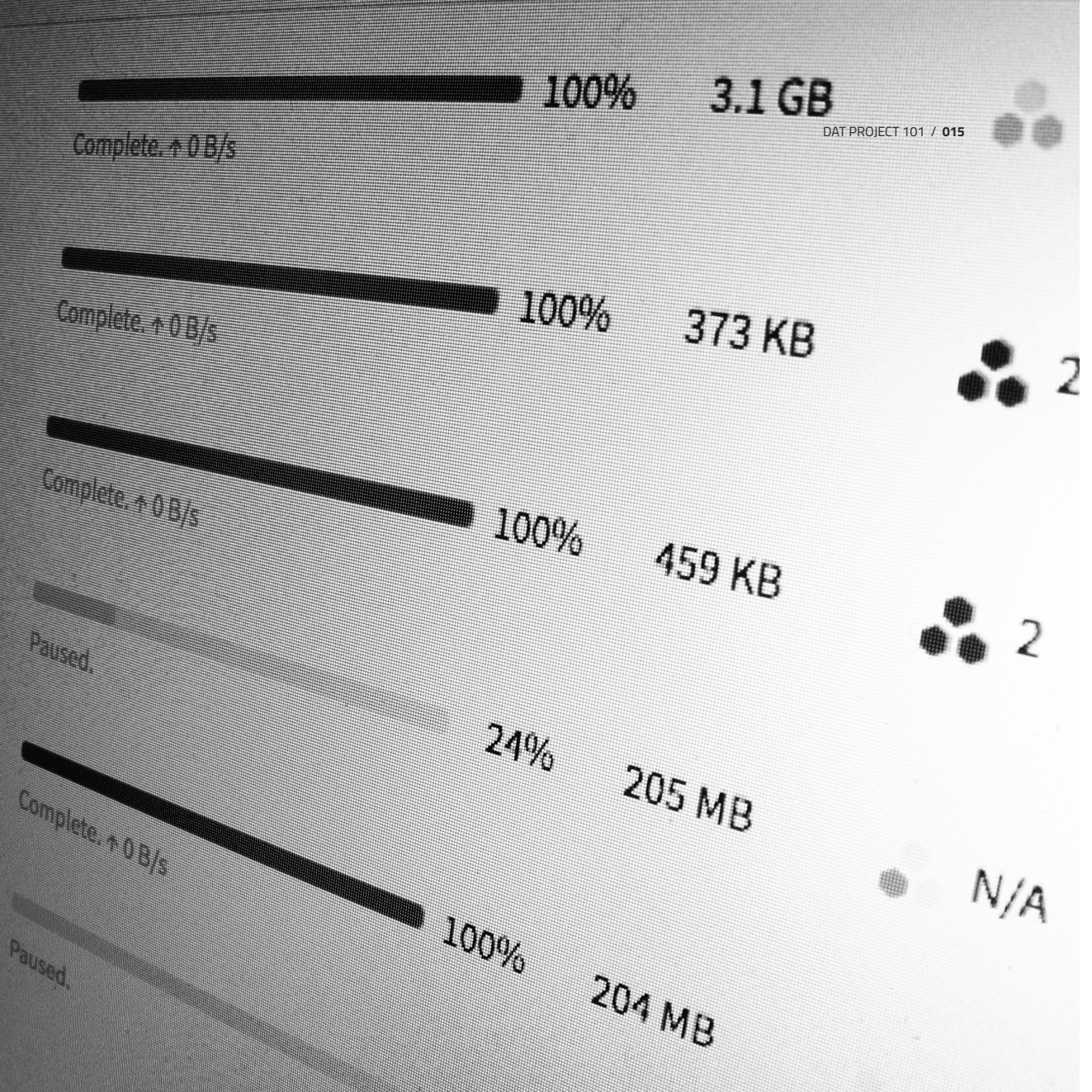
1) Alice has a folder of documents on her system that she wants to share with her friend Bob. She navigates to the directory via her terminal, and issues a `dat share` command to create a new Dat address that corresponds to her directory:

```
$ dat share
Created new dat
dat://7c46dafaa44098d5d439812ed6300
036eaa85bbd9422ee677dbbdc722710a231
Sharing dat: 96 files
```

Behind the scenes, Alice's data is broken up into small pieces, hashed, and arranged into a data structure called a Merkle tree. The resulting data is then easily referenced by a Dat address that looks like this:

```
dat://7c46dafaa44098d5d439812ed6300
036eaa85bbd9422ee677dbbdc722710a231
```

Dat addresses are hexadecimal representations of a public key that corresponds to the shared data. Any user with


100%
3.1 GB
Complete. ↑ 0 B/s
100%
373 KB
Complete. ↑ 0 B/s
2
Complete. ↑ 0 B/s
100%
459 KB
2
Paused.
24%
205 MB
N/A
Complete. ↑ 0 B/s
100%
204 MB
Paused.

the address can clone the data repository, but only the user with the accompanying private key can make changes to it.

2) Alice then sends her newly-made Dat link over to Bob through any standard communication channel. Bob brings up his terminal and runs the following command to clone Alice's data into a directory of his choosing:

```
$ dat clone
dat://7c46dafaa44098d5d439812ed6300
036eaa85bbd9422ee677dbbdc722710a231
alices-data
Created new dat
Cloning: 96 files
```

Behind the scenes, Bob's Dat instance uses a combination of DNS and DHT to connect to Alice or any other peers she may have provided the address to for mirroring. Using two protocols: Hypercore and Hyperdrive, Dat then facilitates connections to machines seeding the files and downloads them.

3) Suppose Alice updates a file within her directory and wants to get that change reflected in her Dat. She can do this easily by rerunning the `dat share` command within her existing directory:

```
$ dat share
```

4) Now, Bob needs to update his local store as well. This can be done with the `dat sync` command:

```
$ dat sync
```

COMPARISONS

Dat is most similar to other decentralized file storing/sharing tools like IPFS and version control systems like Git.

Much like Git, Dat has built in file versioning to track the history of changes made to data. Dats, the repositories, are created and updated very similarly to Git repositories, only without the reliance on a central server.

The concept of addressable, decentralized data sharing is where Dat and IPFS share some overlap. Each protocol allows a user to

create a unique hash to identify data and share it with others to completely clone the data it corresponds to. Dat currently has the added benefit of native browser support with the Beaker browser, allowing the easy creation of Dat-based websites. Websites can be made and hosted in either a user's filesystem with the aid of the `dat` application, or via the Beaker browser itself.

```
https://beakerbrowser.com
```

DOWNSIDES

Dat does have a few downsides worth mentioning. While addresses are touted as secure and non-guessable, they are relatively public and unprotected once found. While a bad actor on the network might not be able to target a specific user directly, there is nothing stopping someone from randomly guessing addresses to see if they correspond to data.

Dat is targeted at the scientific/research community with an aim of sharing public data. If you want to share private data via Dat, it is advisable that you encrypt your data first.

Dat also does not have a global peer swarm. Peers do not come together to join a global network, and instead create smaller, focused networks around pieces of data. This could be considered beneficial as there is less overhead in terms of peer connections, but it also makes it difficult for applications to simply "connect into Dat," leaving applications to implement their own way of interfacing with Dat.

Further, it is important to note that if the original data owner goes offline or closes their client, their Dat address will not work if there are no other peers hosting it. However, once the data is cloned, it will remain on a user's machine even if the original creator disappears from the network. Services like Hashbase (`https://hashbase.io`) offer free and paid storage options for mirroring Dats to keep them online indefinitely.

Besides those points, it's an innovative, easy-to-use tool for storing and sharing data, with many similarities to version control systems.

Written by Mike Dank

BUILDING BEAKER: AN INTERVIEW WITH PAUL FRAZEE

When people think of web browsers, they likely think of FireFox or Brave or any other traditional browser for navigating the World Wide Web. Beaker (`beakerbrowser.com`) is a project that challenges the notion of what a browser can do, and utilizes the peer-to-peer Dat protocol for accessing and building distributed sites and content.

Paul Frazee is the co-founder of the Beaker browser, but he is no newcomer to the world of P2P. Frazee additionally runs a public peer service for the Dat protocol called Hashbase and previously worked on Secure Scuttlebutt, a decentralized, secure gossip platform.

When did you first hear about Dat, and what made you choose this project as a basis for Beaker?

I first heard about Dat while I was working on Secure Scuttlebutt. At the time, Dat wasn't a protocol; it was a command-line tool for working with spreadsheets. Then the Dat project hired Mafintosh to add a P2P networking stack. The idea was, academic researchers often need help syncing files within labs and then keeping published files online, and a bittorrent-style network could kill both birds with one stone. Over time, Dat became better known as the protocol than as the command-line tool.

When I decided to build Beaker, I wanted a files-based protocol. Dat was just reaching its beta state, and so I included it as one of Beaker's browsing protocols. I also included IPFS, but, the more I worked with Mafintosh and their technology, the more I preferred Dat.

One of the reasons for this may seem trivial, but it's not: IPFS has a novel URL design which I couldn't make work with Chromium. Their URL doesn't fit the "standard URL" shape, and

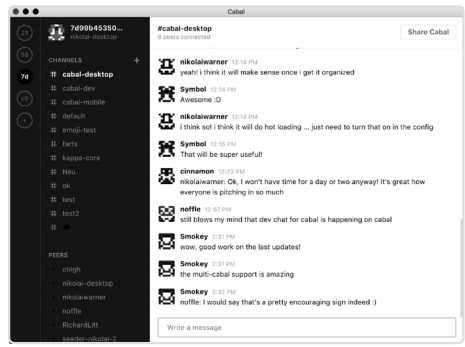
the encoding is case-sensitive which Chromium threw away by lowercasing all hostnames. By the time I was working on Beaker, IPFS was too dedicated to that design-choice to change it. Dat, on the other hand, didn't have a URL design yet, so I was able to work with the Dat team on a URL that would work.

It seems small, but when I had two really good options in front of me, it was like: why struggle with IPFS? Dat was already a solid choice, and it's continued to be great for us.

What things are people out there building with Dat that surprised you?

On Beaker, I was surprised by the creative websites. A lot of what we've seen has been zines, surrealist art, and brutalist design. There's a community of talented Web designers and artists who liked the idea of not needing a server. They've been making some really fun stuff.

On the Dat protocol, Cabal was a really cool surprise. It's a p2p Slack alternative that I'm really interested to see grow.



`https://cabal-club.github.io`

Is there anything you would like to see built on Dat that you haven't yet?

I would really like to see a Facebook killer on Dat. I think the technology can get there, and I'd like for the Web to start fresh on social networking.

One of the things I try to highlight often is that technology has models of authority baked into its design. This is especially true for networks.

On the Web, you're appointing a server to your leader. You have the server controlling the data, the software, and the network itself.

When you don't like something about Facebook, all you can do is say "Oh well, it's just Facebook" instead of going out and changing it. What kind of online society is that? That's not how the Web is supposed to work. That's not how Personal Computing is supposed to work. We should be empowered to change our software.

Peer-to-peer, on the other hand, gets closer to anarchy as a design. That doesn't mean it's chaotic; it just means there isn't any baked-in hierarchy and users can choose leaders for themselves. You can replace your software without having to leave the network, because the software's not controlled by a server. That's how this should work. The design of the social network should be a dialogue within the community that uses it, and the community should be empowered to change it. So, I'd like to see social networking move to P2P, where the users are free to control the experience.

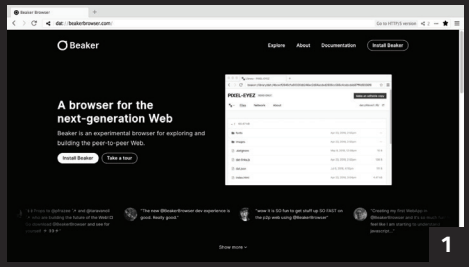
What role are browser like Beaker going to play as the web evolves?

I don't know! There appears to be a market developing around Chromium-based browsers. It's almost like making a Linux distro at this point. There's been Brave, Vivaldi, Beaker, and some others. And now, Microsoft is rebuilding Edge on top of Chrome. That move is *really* going to shape the landscape. It's possible that Chromium forks will become really common, and we'll debate our choices in the same way that Linux users debate their distro picks. Who knows?

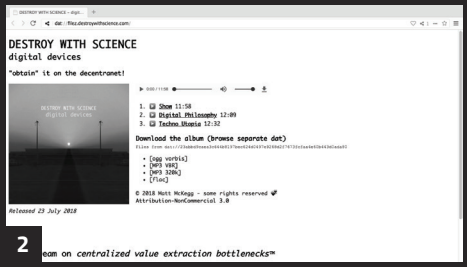
It's hard to say exactly where Beaker sits in the market until we see other vendors react to us. Beaker is meant to be an ongoing experiment, which is why we chose the name. Our mission is to push the Web forward, so success means getting our best features standardized and then moving on to the next experiment.

I guess if we ever run an experiment that no other browser wants to adopt but that our users love then we'll just keep it and that'll be our differentiator.

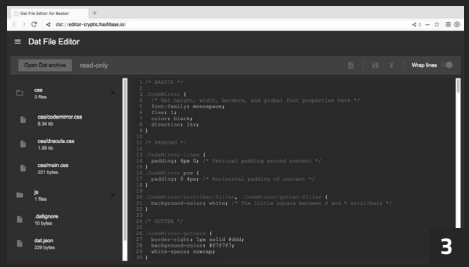
Continued...


Beaker
Explore
About
Documentation
Install Beaker
A browser for the next-generation Web
Beaker is an experimental browser for exploring and building the peer-to-peer Web.
Install Beaker
Take a tour
Show more

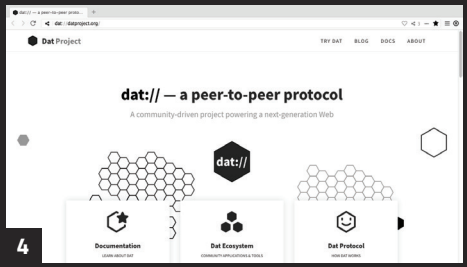
1


DESTROY WITH SCIENCE
digital devices
"obtain" it on the decentranet!
1. Show 11:58
2. Digital Philosophy 12:09
3. Techno Utopia 12:32
Download the album (browse separate dat)
[ogg vorbis]
[MP3 VBR]
[MP3 320k]
[flac]
© 2018 Matt McKegg - some rights reserved
Attribution-NonCommercial 3.0
Released 23 July 2018
eam on centralized value extraction bottlenecks™

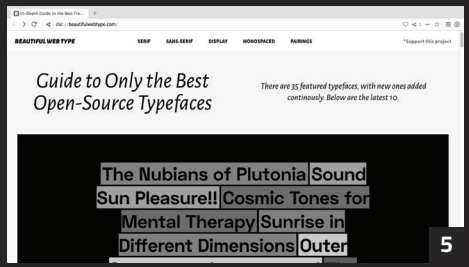
2


Dat File Editor
Open Dat archive
read-only
Wrap lines
/* BASICS */
/* PADDING */
/* GUTTER */

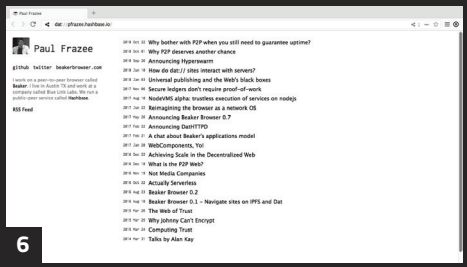
3


Dat Project
TRY DAT
BLOG
DOCS
ABOUT
dat:// — a peer-to-peer protocol
A community-driven project powering a next-generation Web
dat://
Documentation
Dat Ecosystem
Dat Protocol

4


BEAUTIFUL WEB TYPE
SERIF
SANS-SERIF
DISPLAY
MONOSPACED
PAIRINGS
Guide to Only the Best Open-Source Typefaces
There are 35 featured typefaces, with new ones added continously. Below are the latest 10.
The Nubians of Plutonia Sound Sun Pleasure!! Cosmic Tones for Mental Therapy Sunrise in Different Dimensions Outer

5


Paul Frazee
github twitter beakerbrowser.com
RSS Feed
Why bother with P2P when you still need to guarantee uptime?
Why P2P deserves another chance
Announcing Hyperswarm
How do dat:// sites interact with servers?
Universal publishing and the Web's black boxes
Secure ledgers don't require proof-of-work
NodeVMS alpha: trustless execution of services on nodejs
Reimagining the browser as a network OS
Announcing Beaker Browser 0.7
Announcing DatHTTPD
A chat about Beaker's applications model
WebComponents, Yo!
Achieving Scale in the Decentralized Web
What is the P2P Web?
Net Media Companies
Actually Serverless
Beaker Browser 0.2
Beaker Browser 0.1 – Navigate sites on IPFS and Dat
The Web of Trust
Why Johnny Can't Encrypt
Computing Trust
Talks by Alan Kay

6

When every peer stops sharing a Dat, the data within it stops being available. Do you feel this impermanence is a natural characteristic of the web or something that needs repair?

I think there are cases where you can make the impermanence a feature instead of a bug. I don't think that's the most common case, though. I think we need to solve uptime.

My ideal future is that people own servers at home that run software like Hashbase, so that they can have a "home cloud." I'm not sure yet whether people care enough to make that effort.

What we're doing in the mean time is keeping Hashbase on a FOSS license and making sure that it's easy to deploy. It's not perfect, but we've written it to use SQLite so that you don't have to configure a DB or anything. That way, we know we're not standing in the way of home hosting.

How do you see all of these projects growing in the future? Are there any exciting changes on the horizon?

Mafintosh has been discussing this idea called "progressive deployment" that I think could be really exciting.

The idea stems from the fact that that he's developing a way to deploy online services using public keys, similar to how you publish Dat sites. You connect to the services using their public-key address, and the discovery network helps you deal with firewalls and routers.

That technology has two effects. The first: you can move a service from one device to another without the hassle of DNS, because you can reuse the keypair and so the address doesn't change. The second effect: you can run a service from your laptop because the discovery network solves the routing problem.

So imagine this: you write a service in JS or in Web Assembly, then start the service in your browser. That service would run in the background and handle user requests as usual. Then, once you need that service to handle more users, you can move it to a cloud provider without interrupting it. That's the "progressive deployment" part--you deploy at

home first, then progress to more powerful hosts when you need it.

Progressive deployment is very nice for developers because it makes development easier, but end-users are the real winners because it means they can deploy services without any technical savvy. It'd be no different than starting an app on their phone. That's a good case of end-user empowerment.

All of this is going to push us further into a "dumb cloud" world where consumers either self-host their online software, or they rent generic cloud services to host for them.

That sounds awesome man, I look forward to it, and seeing what you do next with Beaker.

EXAMPLE LINKS

```
[1] dat://beakerbrowser.com
[2] dat://filez.destroywithscience.com
[3] dat://editor-cryptic.hashbase.io
[4] dat://datproject.org
[5] dat://beautifulwebtype.com
[6] dat://pfrazee.com
```

THE DOUBLE DRIVE

From the outside, the Double Drive looks like a normal USB flash drive, but inside it holds a secret, completely separate flash chip.

The device itself comprises of a custom PCB adapter (2mm thick), 2 flash chips salvaged from the widely available Sandisk Cruzer Fit USB drives, a 4 pole, double throw (4PDT) switch (MFS401N-2-Z), a 3D printed case, and an optional keychain lobster clip.

ASSEMBLEY

Assembling the Double Drive is relatively straight forward, you'll just need some soldering skills, and a iron with a fine tip.

You can grab the source PCB and 3D files on the specially made dat folder for this zine (see the front of the publication).

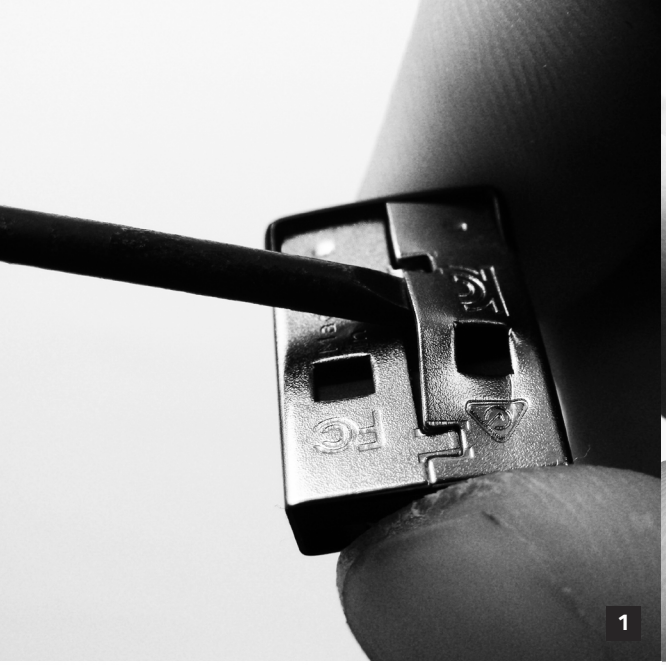
First, use a small flathead screwdriver to open up the Cruzer Fit drives [1]. Make sure to do it from the underside of the USB plug section, but go as far as to pry open the metal shielding. This will lessen the likelihood of you accidentally breaking the chip inside.

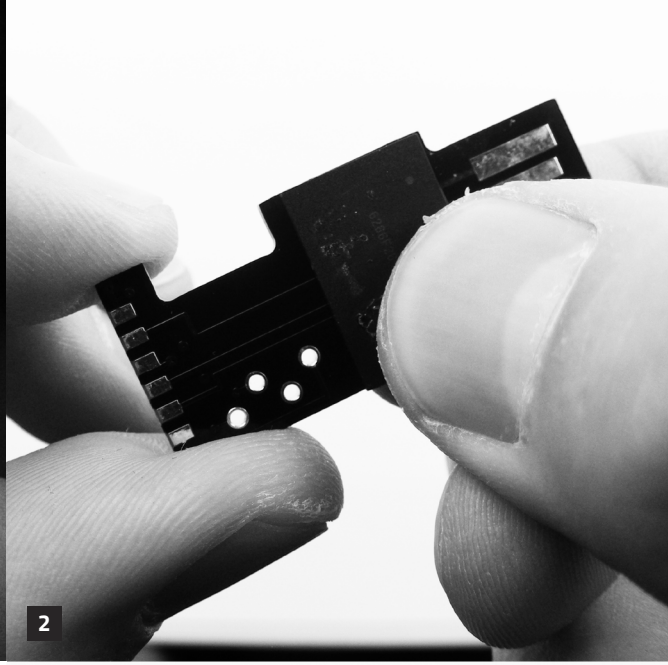
Once released, you can simply line up the metal contacts of the flash drive with the holes on the adapter board [2] (You can add a little glue to stop them moving around before soldering). Make sure the contacts are flat right up against the holes.

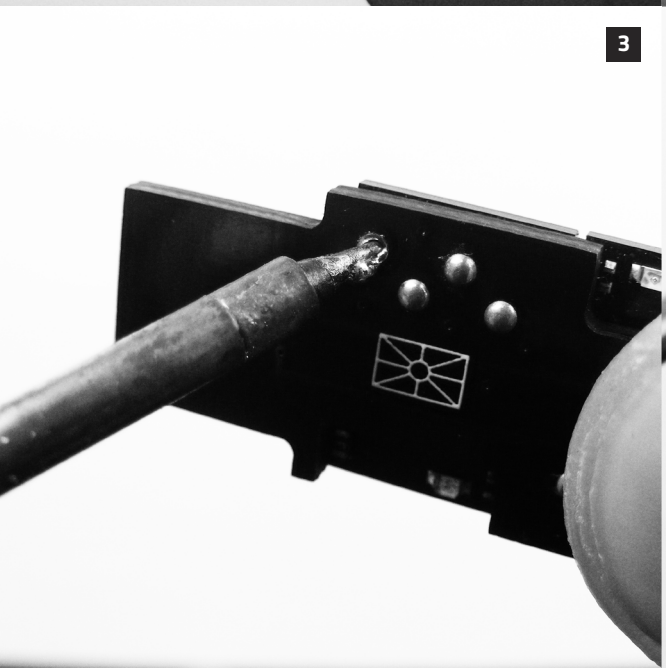
Now you need to solder them on. The best technique here is to add initial blobs of solder to the hole contacts, and then while the solder is still molten, poke the tip of the iron through each hole [3]. This should ensure a good connection is made between the PCB adapter and flash chip.

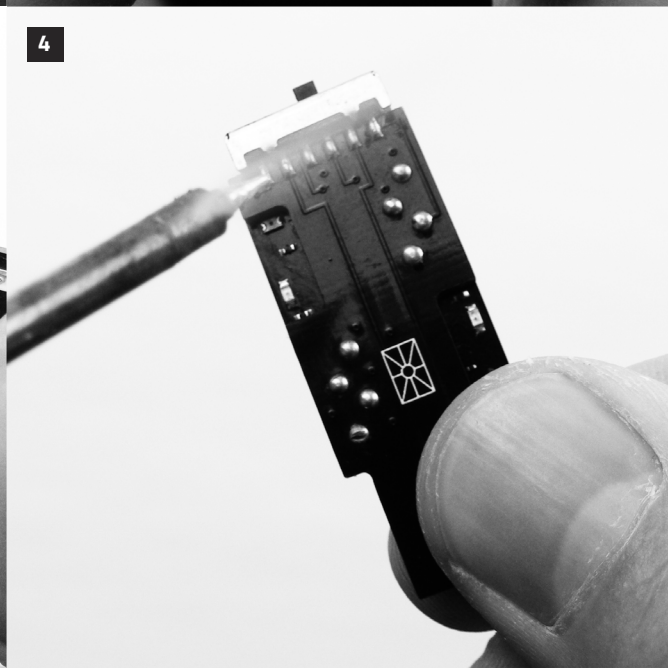
Once both chips are on, you now need to add the switch to the rear of the device [4]. Be careful not to touch the flash chips.

Time to test everything. Place the Double Drive in your computer's USB port [5], and one of the flash chips should light up. Good, one works. Now eject the disk, flip the switch, and reinsert it. The other flash chip should initiate.

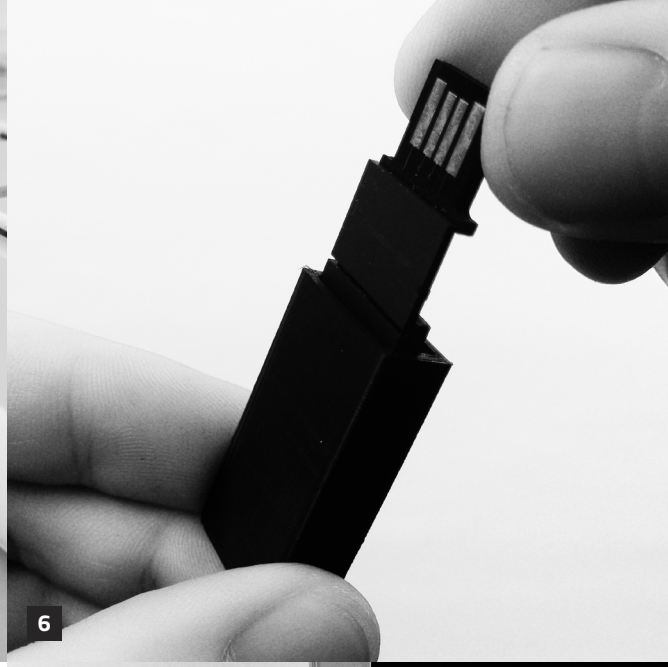

1


2

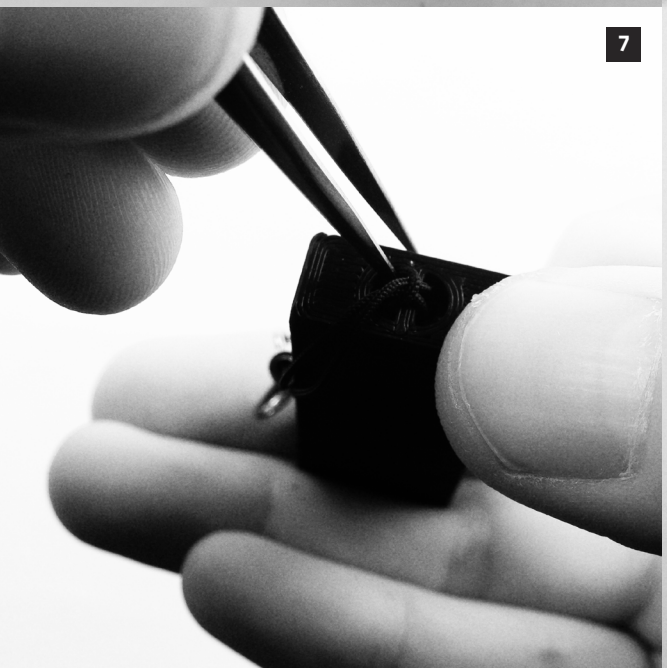

3


4

5

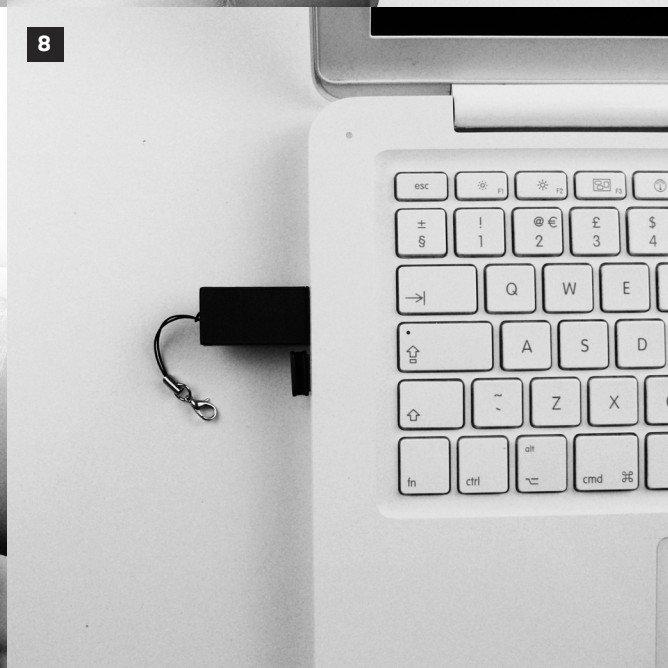


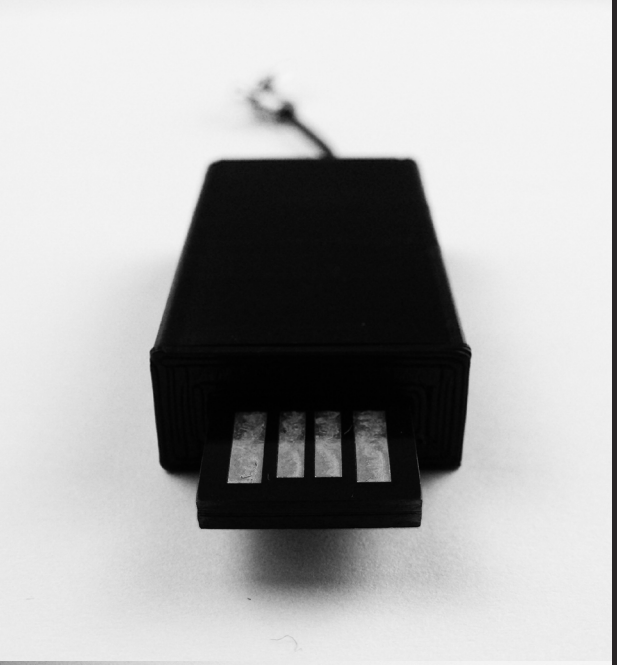
6



7

8





If either don't work, go back and resolder the PCB holes again, and this should bridge any gaps in the contacts.

WRAPPING UP

Alright, if everything's working, just add the keychain clip to the 3D printed case, then assemble everything [6]. You can choose to glue bottom parts of the case together, but that's up to you. Extra stealth points if you finish and paint the case so it looks even more inconspicuous.

You now have a stealth Double Drive. To the casual observer, it appears like a regular drive.

The keychain clip intentionally obscures the back of the device, but if you stick a thin object into the hole [7], you can flick the switch over. Plug it into a computer [8], and you'll see the data on one of the chips only, flick the switch, and you'll get something completely different.

Written by NODE

SO GOD MADE A FARMER

Farms today use a tremendous amount of technology. Where we once had vehicles--cars, trucks, tractors--that were built of analog parts and reparable by any machinist with a lathe and some know-how, we now have machines that are controlled by computers, with software connecting to sensors connecting to engines and actuators. It is a different world, but not necessarily a better one.Manufacturers boast that new tractors are safer and easier to maintain than ever. Software makes sure that everything is working properly, and if something goes wrong, it throws an error alerting to the problem. Unlike with earlier tractors, farmers these days can't simply fix their machines and get back to work if something goes awry. The manufacturer has to be included in the repair loop, and you're left in the dust if you don't want to go through them.

Within the past few years, farmers have joined the Right to Repair movement as their tractors and other equipment have become less and less user-serviceable. For generations of machinery, any person could strip down a tractor to its individual parts, clean them, and build things back up without needing anything but time and patience. These days, when a tractor breaks down, manufacturers like John Deere require machines to be sent away to authorized repair centers, enforcing these rules with software lockdowns and tampering prevention mechanisms that a normal user--even a crafty one--cannot get around without some level of outside assistance.

A whole community of farmers, who have been known to tamper, tinker, and construct makeshift solutions to complex problems, have been left out in the cold by manufacturers of equipment essential to maintaining a livelihood. While a tractor that first rolled off of the assembly line fifty years ago may still be able to put in a full day's work, who knows the fate of the current generation coming out of the showroom. Will future fixes be impossible if authorized service centers refuse to support older equipment? In today's culture, a lot of technology is considered disposable--longevity isn't factored in when

Repairing Tractors The Old Way
Photo by FOTO:FORTEPAN / Mészáros Zoltán (Creative Commons 3.0)

pushing out a new product. Planned obsolescence reigns supreme.

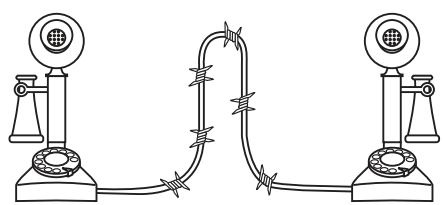
In the US, farmers have been granted an exception to the Digital Millennium Copyright Act (DMCA), a piece of 1998 legislation that enforces intellectual property rights and criminalizes circumvention of access control. However, manufacturers retaliated by updating their terms of service to disallow the tampering that this law would permit. Though it is fully legal for farmers to make certain repairs, it is against company policy, and manufacturers don't need to make things simple, or have materials available to assist. As with companies like Apple and Microsoft and their respective iPhone and Xbox hardware, these manufacturers don't want anyone looking into the internals of a product, even after it becomes a customer's own property. Luckily, legislation is changing, albeit slowly. Aligning with smaller strides in the techno-legal space such as the unenforceability of "Warranty Void If Removed" stickers, tractor manufacturers have agreed to produce repair manuals, firmware updates, and replacement parts in California by 2021, but with no currently outlined plans for pricing or delivery mechanisms. Currently, a farmer can obtain a replacement part from a third party and even install it, but the machine won't run until an authorized repair center locks the part to the tractor via a firmware update that marries their serial numbers. DRM-culture has infected many aspects of daily life, and it can only cripple technological advancement as time goes on.

Farmers have always been self-reliant: they take repairs and construction into their own hands to save time and money. When you need to get things done on a schedule, there isn't a lot of time to waste in getting things up and running again; why spend thousands of dollars and send a machine away when a quick fix will do? Numerous websites exist to support farmers who wish to share their designs and inventions, including `https://farmhack.org/tools` and `https://www.opensourceecology.org`. Historically, there have been many interesting technological feats accomplished by farmers.

One such marvel is the introduction of barbed wire telephone lines going back to the 1880s.

https://farmhack.org/tools

When telephone companies wouldn't extend infrastructure out to farmers, ranchers, and other homesteaders, they banded together to extend service from one house to communities of 20 or more residences by wiring up pairs of barbed wire lines, already deployed on fences, extending out around properties. While the barbed wire was initially erected to contain livestock, clever minds thought it could also be purposed to carry communications to phones set up in homes and barns that the telephone companies wouldn't touch.

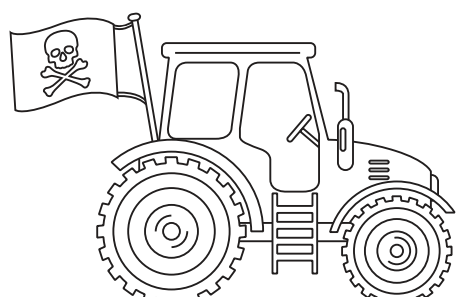


While many of these configurations may have been crude, they were simple, convenient, and inexpensive. Some groups may have had all phones ring when the number was dialed (relying on agreed-upon codes, like three short rings for you and two long rings for me), while others contained extensive switchboards manned by volunteers.

A more recent analog we see if the rise of community broadband networks in rural areas. Initiatives such as B4RN in Lancashire, England and RS Fiber in Minnesota, USA were founded through grassroots efforts to combat a lack of high-speed Internet availability. Some farmers and ranchers got fed up and started trenching and lighting their own fiber, while others started deploying wireless access points and backhaul links to cover area ISPs wouldn't service. Many of these installations are done through member-owned cooperatives set up specifically for broadband expansion within communities. These built-outs are often performed without federal funding, and ensure that communities have Internet service essential to their well-being and businesses.

When you really stand back and look at everything, it is easy to conclude one thing: farmers are hackers by nature. They tinker and improve and figure things out for themselves. Hacker culture itself can be traced back to the hippie movement of the 1960's. In the USA, hippies were banding together to create communes on patches of land that could be run in a self-sufficient manner.

These homesteaders would carry out agricultural tasks, construction, infrastructure planning, and basic jobs to contribute to the livelihood of their community. Merry Pranksters, Whole-Earthers, Douglas Englebert disciples, and engineers blended in high collaboration. Steve Jobs, Ted Nelson, and Xerox PARC pioneers all drew from this self-sufficient culture, creating an explosion of mind-expanding technological inventions that would go on to shape our present.



Today, the hacking continues. When faced with vendor lockouts on modern farming equipment, farmers are using pirated software originating out of Eastern Europe to get their machines up and running again.

Private, invite-only forums sit innocuously on the web where farmers can go to get cracked software for their machines. After paying a membership fee, black-market software kits, keygens, and cracks for the latest tractor models are all available--for the right price.

Without knowing it, farmers have become activists, freeing their property from the stronghold of manufacturers and teaching others to do the same. Nobody is quite sure what will happen in California come 2021 with regards to promised parts and software, but many are worried that this agreement might hinder full Right to Repair legislation being passed in certain states further down the road. Because lawmakers and interest groups settled on a small solution now, it'll be harder to ask for more later.

Manufacturers are feeling the heat, but no matter what happens, we can be sure of something that has held true for generations: farmers will take things into their own hands.

Written by Mike Dank

LIKE IRC OVER RADIO WAVES: CHATTERVOX TUTORIAL

Packet radio is a technology that has existed for decades, going back to projects like ALOHAnet in 1971 where it was used to transfer data through the Hawaiian islands.

Based on the simple idea of transmitting digital data over wireless communications, packet radio is used today by both radio amateurs and commercial organizations, such as those working with dispatch networks and even cellular phone deployments.

In the amateur radio space, many countries ban encrypted communications of any nature. While Chattervox does not offer a way to encrypt conversations over the radio waves, it builds upon AX.25, a popular packet radio chat protocol supporting digital signatures and binary compression. Chattervox adds a chat implementation on top of AX.25, and includes elliptic curve cryptography to allow users to sign and verify messages with one another in real time, even when they are miles apart.

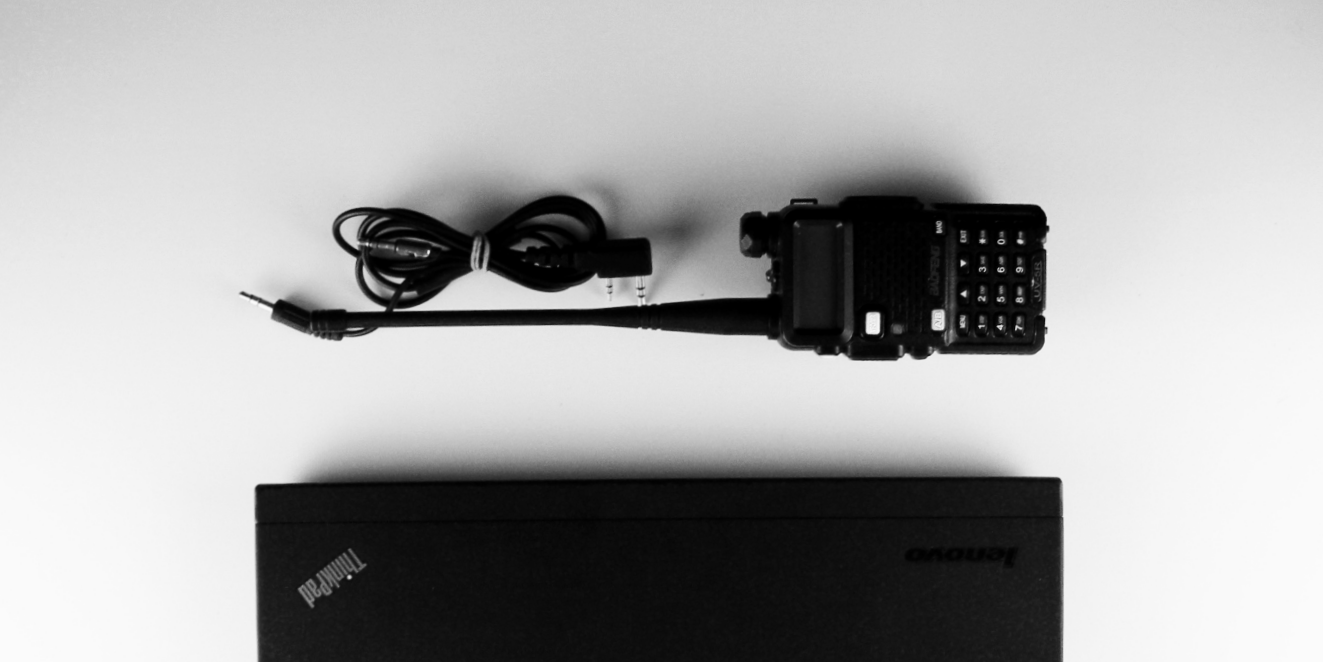
As of this writing, the software is in beta and relatively new. That said, the current version is stable, easy to use, and simple to set up with inexpensive hardware. Please note, depending on your country of residence, you may need to acquire an amateur radio license to use Chattervox legally depending on what frequency band you choose to use.

MATERIALS NEEDED

I will assume that you already have a computer with a Unix-like operating system and a 3.5mm TRRS jack for audio in/out (or separate line-in/line-out jacks with an appropriate adapter). I will be using a generic laptop running Debian Linux. You'll also need:

- Baofeng UV-5R 4-watt radio ($19.99 USD). A very inexpensive but capable VHF/UHF radio.

- BTech APRS cable ($18.79 USD). A 3.5mm TRRS to 3.5mm and 2.5mm audio cable to interface with our radio.



Alternatively, you can make the cable yourself with instructions from the zine, “Messing Around With Packet Radio” by O’Really. You can download it for free from Archive.org.

SOFTWARE SETUP

First, we need to log into our system with a non-root, sudo user.

Debian's package repositories may not have an up-to-date version of Node.js included, so we may need to install it separately from the NodeSource Node.js Binary Distributions page, `github.com/nodesource/distributions/blob/master/README.md`

This guide has been tested with `node` v11.0.0.

```
$ node --version
v11.0.0
```

Continued...

After we verify that Node.js is installed, we can install Chattervox via `npm` (all on one line):

```
$ sudo npm install --cli -g
chattervox@latest
```

Before starting Chattervox, we will install a popular software-based TNC substitute known as Dire Wolf:

```
$ git clone
https://github.com/wb2osz/direwolf
$ cd direwolf/
$ sudo apt-get install libasound2-dev
$ make
$ sudo make install
$ make install-conf
```

RUNNING CHATTERVOX

Now that all of the software is installed, we can hook up our radio to our computer using the APRS cable, and make sure the radio is turned on and in VOX mode.

Anyone participating in the chat will need to tune in to the same frequency. For basic communications, a simple simplex (radio-to-radio, not using a repeater) frequency for packet radio can be used, such as 147.027 MHz on VHF.

We can now start `direwolf`:

```
$ ./direwolf -p -q d -t 0
Dire Wolf version 1.5

Reading config file direwolf.conf
Audio device for both receive and
transmit: default  (channel 0)
Channel 0: 1200 baud, AFSK 1200 &
2200 Hz, E+, 44100 sample rate.
Note: PTT not configured for channel
0. (Ignore this if using VOX.)
Ready to accept AGW client
application 0 on port 8000 ...
Ready to accept KISS TCP client
application 0 on port 8001 ...
Virtual KISS TNC is available on
/dev/pts/3
Created symlink /tmp/kisstnc ->
/dev/pts/3
```

`chattervox` can then be started using the `chat` flag. On the first run for a user, it will set

up your identity and public/private keys, saved to `~/.chattervox`.

```
$ chattervox chat
Welcome! It looks like you are using
chattervox for the first time.
We'll ask you some questions to
create an initial settings
configuration.

What is your call sign (default:
N0CALL)? N0CALL
What SSID would you like to associate
with this station (press ENTER to
skip)?
Do you have a dedicated hardware TNC
that you would like to use instead of
direwolf (default: no)? no
{
  "version": 2,
  "callsign": "N0CALL",
  "ssid": 0,
  "keystoreFile":
"/home/user/.chattervox/keystore.json
",
  "kissPort": "/tmp/kisstnc",
  "kissBaud": 9600
}
Is this correct [Y/n]? Y
Generating ECDSA keypair...
Public Key: 04cf3a8d53fb6f8fce248f...

Settings saved to
/home/user/.chattervox/config.json
N0CALL:
```

Just like that, we're in chat and able to communicate with a anyone else on the frequency with the same setup.

If you are having trouble communicating, try adjusting the squelch setting on your radio to '1' and make sure audio is working via your computer. Keep in mind, Baofeng radios may transmit for an additional second after the output signal has concluded while in VOX mode, so you may miss some RX in that time!

KEY EXCHANGE

Chattervox really shines when you use it for digitally signed messages, meaning you can verify someone sending a message is who they say they are. We can view our own keys using the `showkey` option, which prints the


lenovo
X200
VFO/MR
A/B
BAOFENG
BAND
MENU
EXIT

public key that we can send out to contacts (don't share the private key with anyone):

```
$ chattervox showkey
N0CALL Public Key (your signing key):
04cf3a8d53fb6f8fce248f5249bc2...
N0CALL Private Key: <REDACTED>
```

We can import keys from our friends (assuming they were sent over secure channels) using the `addkey` option, including a callsign and a public key as arguments:

```
$ chattervox addkey KC3LZO
04cf3a8d53fb6f8fce248f5249bc2...
```

Verified contacts will show up in chat as normal, but others will display info that might make you second-guess their validity:

```
N0CALL: What's up everyone?
KC3LZO: Not much!
J29DBA (KEY NOT FOUND): Hey it's me
Steve, good to be here for the first
time.
K4X (INVALID SIGNATURE): It's Bob!
How is everyone?
N2ASD (UNSIGNED): Alive and well!
```

CONCLUSION

That's all it takes to set up and use Chattervox on commodity hardware. In theory, Chattervox can be used via other communication methods, including unlicensed radio bands, or even via sneakernet for slow communications where audio is physically recorded, transported, and played back at the intended recipient's location.

You can learn more about Chattervox by visiting `https://n-o-d-e.net/url/3/`.

Also, if you wish to share your public key with a wide group of Chattervox users, there is a public key repository you can contribute to, `https://n-o-d-e.net/url/4/`

Written by Mike Dank

RICOCHET: INSTANT MESSAGING ON THE TOR NETWORK

Ricochet (or Ricochet IM), originally released in 2014, is an open-source instant messaging platform focusing on privacy. Unlike most direct peer-to-peer chat applications, Ricochet utilizes the Tor anonymity network to facilitate connections.

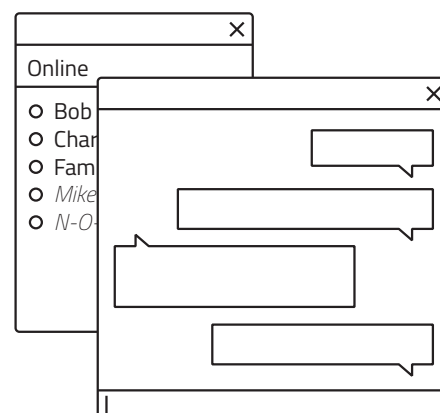
HOW IT WORKS

Ricochet's key features include its anonymized connection layer and its unique authentication system based on public-key cryptography.

Ricochet addresses look like this and are derived from the address of a Tor hidden service launched by Ricochet on the user's machine: `ricochet:pq3axign4v4nlpq5`

When a Tor hidden service is set up, a public/private key-pair is generated for it to use.

A hidden service's address is derived from the public key, while the private key is kept secret by the service operator. If the private key and public key don't cryptographically match, the Tor hidden service will not function. This architecture makes impersonating hidden services, or the operators behind them, virtually impossible due to the cryptographic complexity involved with forging a new private key to match the exisiting public key.



To illustrate how it works, here is what happens when Alice wants to message to Bob:

1. Alice and Bob exchange Ricochet addresses through an existing, trusted, communication method such as an in-person meetup.

2. Alice chooses to add a new contact and inputs Bob's Ricochet address, optionally adding some message text before pressing "Add."

3. In the background, a connection is made between Alice and Bob over the Tor network, sending Alice's contact request along with authentication information about her hidden service. Since this connection is made from Alice to Bob's hidden service, it is signed and encrypted with Alice's private key and Bob's public key. This means that the request can only have been sent by Alice, and can only be read by Bob.

4. Bob now sees that he has a new contact request. Ricochet has already verified that the request has come from Alice's hidden service, and Bob knows Alice, so he accepts it.

5. Behind the scenes, Alice's and Bob's machines open a new connection and keep it open to accurately determine if the other is online at any given time.

6. Now, they will each appear as online in each other's contact lists within Ricochet. Alice or Bob can now instigate a new message with the other over the encrypted, anonymous connection that runs between them.

Most of the technical aspects of this transaction happen in the background and are hidden away from the user. To anyone using Ricochet, the process of messaging a contact resembles that of any other instant messaging application.

COMPARISONS

Compared to traditional and other decentralized instant messaging applications, Ricochet has a few advantages.

	Google Hangouts	*Tox*	*Ricochet*
Design	Centralized	Decentralized	Decentralized
E2E Encryption	No	Yes	Yes
Anonymous	No	No	Yes
Location	Visible	Visible	Hidden
Group Messaging	Yes	Yes	No
Voice Messaging	Yes	Yes	No
File Transfer	Yes	Yes	No

For one, all messages on the platform are end-to-end encrypted via public-key cryptography and authenticated, so they are completely private between the sender and receiver.

The location of the sender and receiver are also completely hidden via the Tor network. While other applications may let you connect through Tor, not all of them have it as default.

Ricochet is also a decentralized application, not relying on or trusting any central services or operators. Connections between users are done P2P and do not require any additional messaging infrastructure.

Lastly, Ricochet has undergone a security audit by the NCC Group, with positive results.

DOWNSIDES

Being decentralized, Ricochet does not have any concept of registration or login, so an attacker with physical or remote access to a

user's machine could assume that user's identity (though this is a general challenge with all P2P software).

Much like with Bitcoin addresses, users need to perform their own backups in the event that their machines fail. Without a backup, Ricochet users would have to create a new address and identity, then perform all of their contact requests a second time.

Lastly, even though Ricochet uses a binary protocol for communication, it currently only supports text messaging. There are, however, currently plans for file transfer in the future.

CLIENTS

The official Ricochet GUI client is available for Mac, Windows, and Linux, and downloadable as a binary from the Ricochet.im site. There is also the option to build Ricochet from source.

```
Ricochet
https://ricochet.im
```

Alternatively, there is an unofficial ricochet-irc application which runs Ricochet acting like an IRC server to facilitate access through the terminal if preferred.

```
ricochet-irc
https://github.com/wfr/ricochet-irc
```

The Ricochet team is currently working on a new implementation of the protocol in the Go programming language and appears to have plans for new GUI and CLI versions of the Ricochet client.

```
https://github.com/ricochet-im
```

CONCLUSION

So that's the basics of Ricochet. It has the benefit of secure communication between users, and it is completely decentralized and trust-less. Additionally, the application's use of the ever-growing Tor network helps to enhance privacy for the users.

Written by Mike Dank

DIGITAL FINGERPRINTING & TRACKING

Your digital fingerprint consists of the habits, hardware, and the software you use to interact with the Internet.

The more points of data that can be gleaned from a person, the more unique the fingerprint is, and the easier it is to identify and track someone, even if they think they're acting anonymously by doing something like masking their IP address.

Here's an overview of some of the ways your personal fingerprint is constructed. It applies to both browser-based tracking, as well as tracking done through other apps.

(Further fingerprinting techniques are always being developed, so this list is not exhaustive.)

JS Navigator. The navigator object allow websites to know the basic makeup of your system, and is often used as the most basic form of tracking. It allows websites to see your browser type, the version number, your operating system and version, your CPU type, screen resolution, color depth, language settings and more.

HTML5 Canvas. The Canvas API is a feature in HTML5 which is used to draw 2D graphics and animations on web pages. It fingerprints users by drawing a hidden canvas, and converting that to a hash. Since every computer is slightly different, this creates a unique identifier, which can be used to track you.

WebGL. Again, similar to Canvas, this fingerprinting technique deals with how your system renders 3D graphics for web pages. The combination of specific hardware and drivers will result in a different, trackable hash.

Timezone. Websites can find your time zone in two ways. First by using your IP address, and second through Javascript functions which read the regional settings of your operating system. If you use a proxy or VPN, this second method can reveal your real time zone, and the discrepancy between your real time zone and the proxy time zone will make you stand out further.

Audio Context. Similar to HTML5 Canvas, the AudioContext fingerprint is based on your system's physical hardware capabilities. In this case, the unique hash relates to your audio stack and settings.

Browser Storage. Browsers store a variety of local data related to your browsing habits, such as cookies, browser history, extensions, settings, local storage and more. Depending on your browser and personal settings, websites may have open access to this storage, making it a good way to track you.

WebRTC. This plugin, which allows for P2P web applications through the browser, can leak both your public and local IP addresses, even if you're using a proxy.

Fonts. If you've installed custom fonts on your system, these can be read by websites, making your fingerprint much more unique.

Media Devices. Another feature of WebRTC is that in order to successfully connect users through P2P connections, a hash of the media devices, such as cameras, microphones and audio hardware, is required. This is called Device ID, and it can be used as a way to identify users across the web.

Battery Level. The Battery Status API allows for over 14 million different values for a battery's current state, giving a very unique fingerprint. This can allow ad tracking scripts to track the same person over completely unrelated websites.

Plugins. Similar to fonts, if you have custom plugins installed, especially old versions, then this will make your fingerprint more unique.

Usage Patterns. If you visit the same handful of websites every day, at the same times, then you have a trackable pattern. If the traffic is unique enough compared to other users, it means even if you changed a variable, like your location, or hardware/software used, you could still reasonably be identified.

Wifi Router in Range. If an app has the correct privileges, it can analyze which Wifi access points are in range of your device. When combined with other data points from other users, this can track your location, even if you're moving about, offline, or switching between IP addresses.

Consolidation. If a company buys another, and you've used both of their products, those data profiles can be consolidated.

Ultrasonic Sounds. Some ad trackers have been found to emit unique codes through ultrasonic sounds, which are outside of human hearing range, but can be heard by other phones or computers. This means other devices in the area running the same ad tracking scripts can pick up the code, establishing another data point and tentative connection to your fingerprinting profile.

Keyboard/Mouse Patterns. You may not realize that you have a certain keyboard typing cadence or quirk to how you move a cursor on a screen, but JavaScript tracking scripts will be able to record this for analysis.

File Metadata. When you upload files to online services, the metadata in them can be analyzed and stored. For example, photos and videos can store data about the camera or phone used to take the picture, the GPS coordinates where they were shot, the date and time of creation, and lots more.

Facial Recognition. It's not just file metadata though, the actual content can be tracked, especially with faces.

If you upload photos to large companies like Facebook or Google, or if you use services which utilize their APIs, this face data is stored. Even if you randomly appear in the background of a stranger's photo, the combination of metadata can still link you to people, locations, usage patterns, etc.

Hardware Fingerprint. Apps installed on your devices have greater control over what data is gleaned, so they can see exactly what hardware you are using. If you a have non standard, custom setup, this will set you apart from others.

Both your computer, and network interfaces have unique identifiers (serial number and MAC addresses), meaning if some apps are made by the same company, or use the same APIs, they can identify you.

File Fingerprinting. When you download files from apps or websites, they may insert unique data into them which can be tied to you, so if someone else uploads the same file down the line, the service will know where it came from, creating a new data point.

Movement Tracking. Even if you have location tracking turned off for your phone, the accelerometers can see when you're up and about, and create a movement profile, so phone manufacturers or app developers can still track you over different devices.

CONCLUSION

These are just a handful of the ways we're fingerprinted and tracked online. It's impossible to completely get rid of your unique fingerprint, but you can aim to blend in with the crowd.

Written by NODE

PI-HOLE ON THE NANO SERVER: BLOCK ADS & TRACKERS

This short guide will show you how to install Pi-Hole (`https://pi-hole.net`), a network-wide ad blocker, on a NODE Nano Server.

Obviously, this will also work fine on a regular Pi, but the compact size and always-on nature of the Nano Server makes it a perfect match.

This allows all devices on your network to get the same ad block features, just by changing DNS settings on your devices.

SETTING UP THE PI

First, we have to download and burn the latest lite version of Raspbian to a micro SD card. If you're a beginner, visit:

`https://raspberrypi.org/downloads`

Next, we need to set up the WiFi connection. Insert the micro SD card you just burned Raspbian to into your computer (not the Pi).

Create a blank text file, name it `ssh`, then drag it into the micro SD volume's root directory.

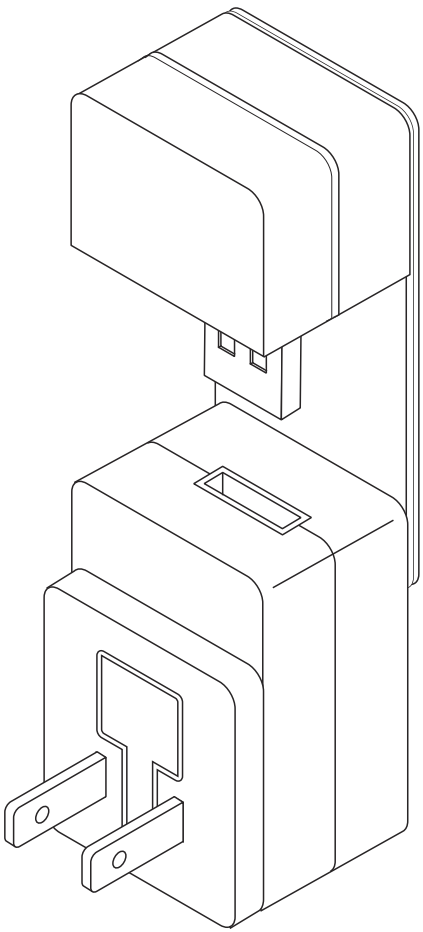
Now create another text file, and name it `wpa_supplicant.conf`. Add the following text, replacing the SSID and Password with your WiFi details:

```
country=us
update_config=1
ctrl_interface=/var/run/wpa_supplicant

network={
 scan_ssid=1
 ssid="YourWifiSSID"
 psk="PasswordGoesHere"
}
```

Save it, and drag that into the micro SD volume too. Eject the micro SD card, and insert it into the Zero W.

When you turn on the Pi for the first time, it'll connect to your WiFi, and SSH will be enabled



INSTALLING PI-HOLE

Installing Pi-Hole is super simple, just open a terminal, and SSH into your Zero W, replacing `192.168.1.10` with your Pi's IP address:

```
ssh pi@192.168.1.10
```

The password is still set as the default `raspberry`, so we should change that too. Type `sudo raspi-config` into the terminal, press Enter, and change your password.

When that's done, run the Pi-Hole installer by typing the following (all on one line):

```
curl -sSL
https://install.pi-hole.net | bash
```

This automatically downloads and sets everything up for you. It will ask you a few questions about how you want your system to be configured, so pay attention. Once finished, it'll tell you what the address and admin password are.

You now can unplug the Zero W, screw it into the Nano Server, and plug it into the outlet.

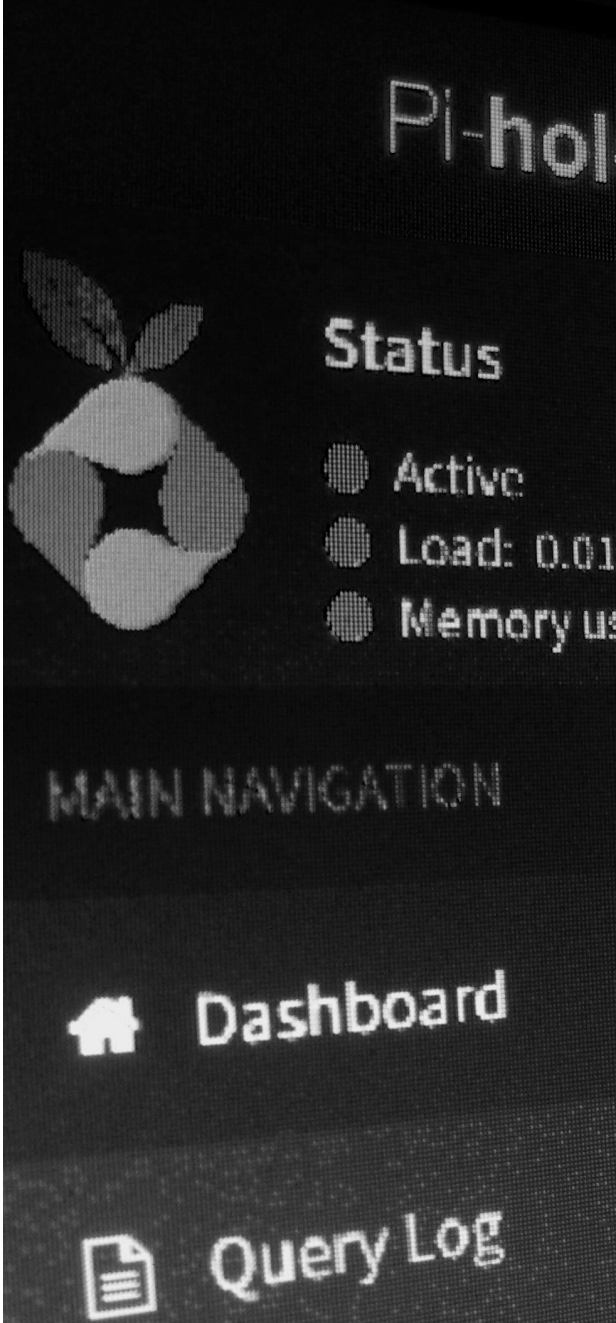
BLOCKING ADS & TRACKERS

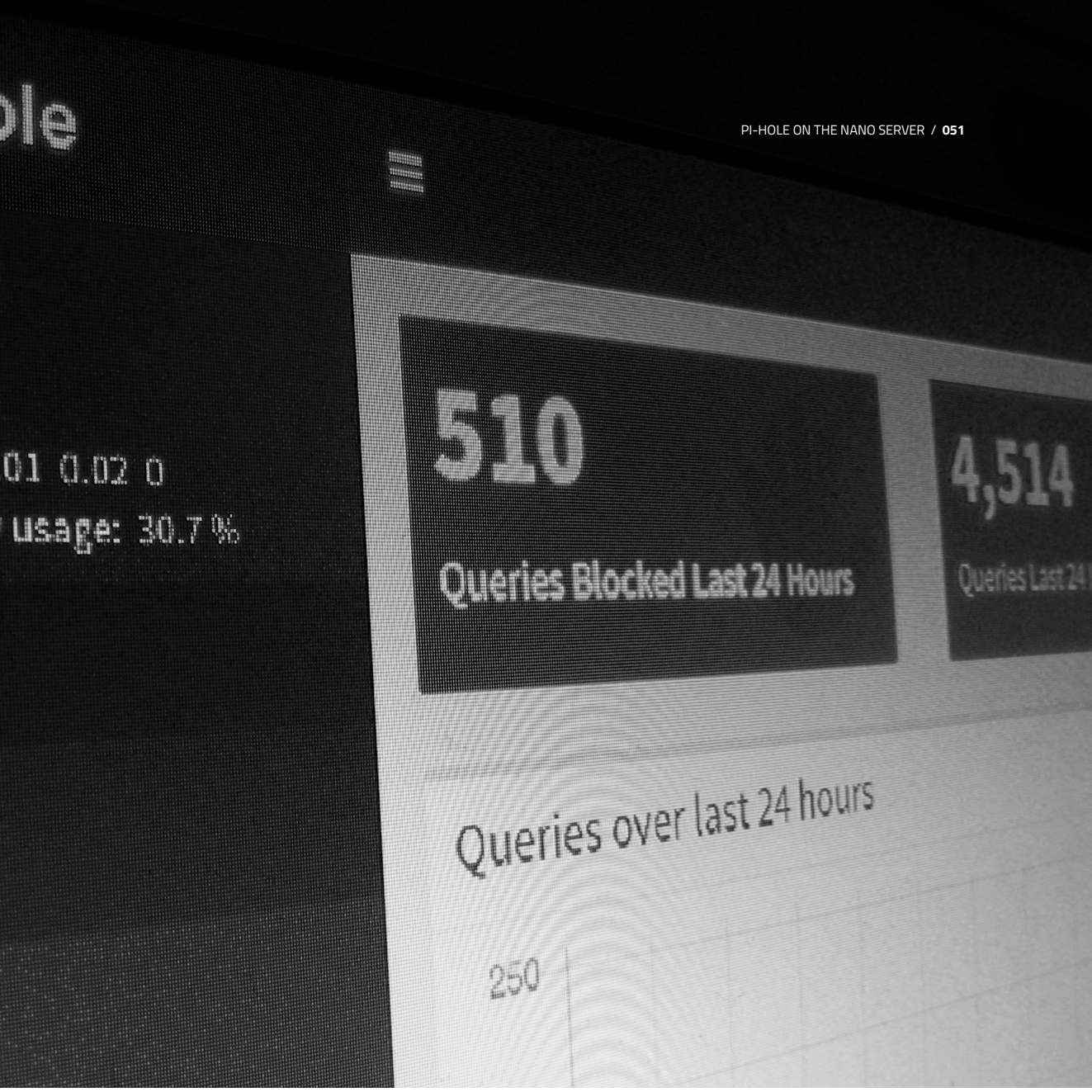
To get the Pi-Hole working on your computers, tablets, and phone, you simply change the DNS settings on your device by adding the Pi Hole's IP address (the one it gave you earlier). You may need to restart your browser, but ads should automatically be blocked from there onwards.

If you play around in the admin interface, you can check out the realtime stats for blocked requests, set up whitelists and blacklists, and do all sorts of other useful configuration. This is a great way to mitigate ads and tracking on a wide range of devices. Obviously it's not going to catch everything, but it's a good start.

If you're more advanced, also check out `https://pi-hole.net` for guides on how to set up a VPN on the device, allowing you to use it while you're away from your home network.

Written by NODE




ole
01 0.02 0
usage: 30.7 %
510
Queries Blocked Last 24 Hours
4,514
Queries over last 24 hours
250

LIBREBOOTING THE THINKPAD X200

The problem of increasing hardware and software complexity in modern computers has led to a greater loss of control and freedoms from a user's standpoint. For example, it's safe to say all new CPUs have secret backdoors, and perhaps other nefarious chips running on their motherboards, as well as built in microphones, cameras, and software that spies and tracks users in 1000 different ways.

Enter the ThinkPad X200. This little beast was state-of-the-art when it was released in 2008. It sports a 12.1" 1280x800 pixel screen, an Intel Core 2 Duo 2.40 GHz CPU, 4GB RAM (I believe it can go to 6GB), an excellent keyboard, and all the other features you'd expect from a ThinkPad workhorse. You can easily find them on Ebay going for $50-$100.

What's more interesting is that this model has a separate BIOS chip that you can reflash with Libreboot (`libreboot.org`), to remove any potential boot-related backdoors, and that's what I'm going to show you how to do.

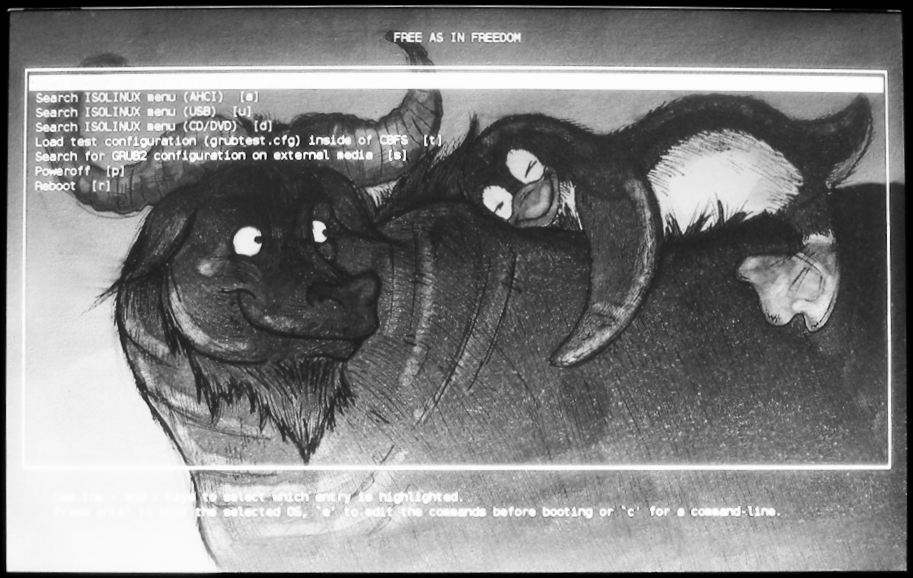
(Shout out to /u/T-400/ on Reddit for bringing together a lot of the info used in this guide.)

MATERIALS

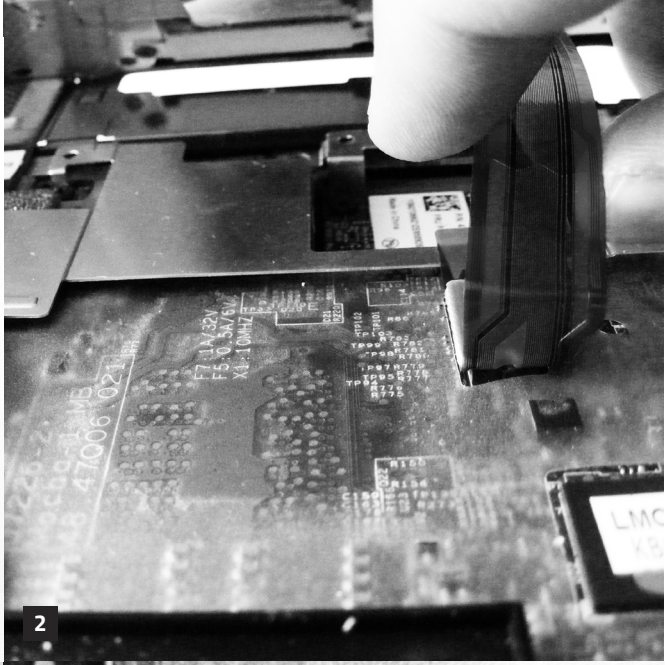
- ThinkPad X200
- Raspberry Pi 2, 3 or Zero
- A SOIC-8 or SOIC-16 clip (depending on the chip you have)
- Female to Female jumper cables (or regular wires if you manually solder them)
- A separate computer to SSH from

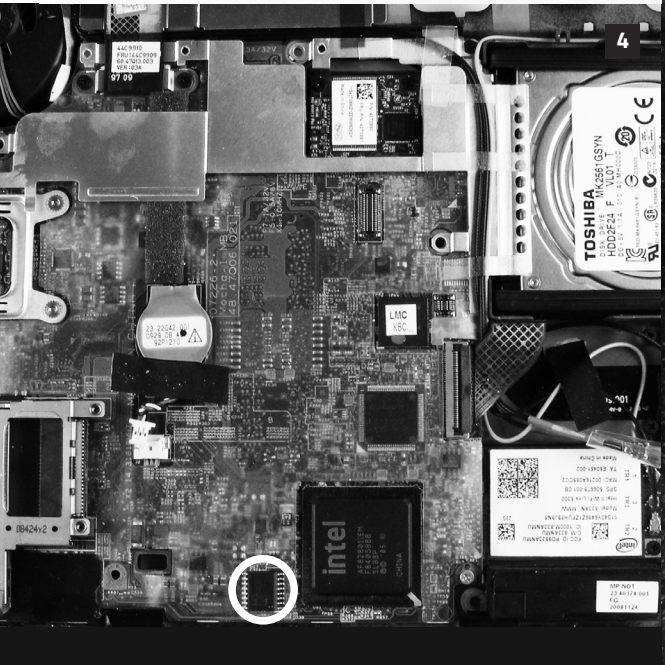
MAC ADDRESS

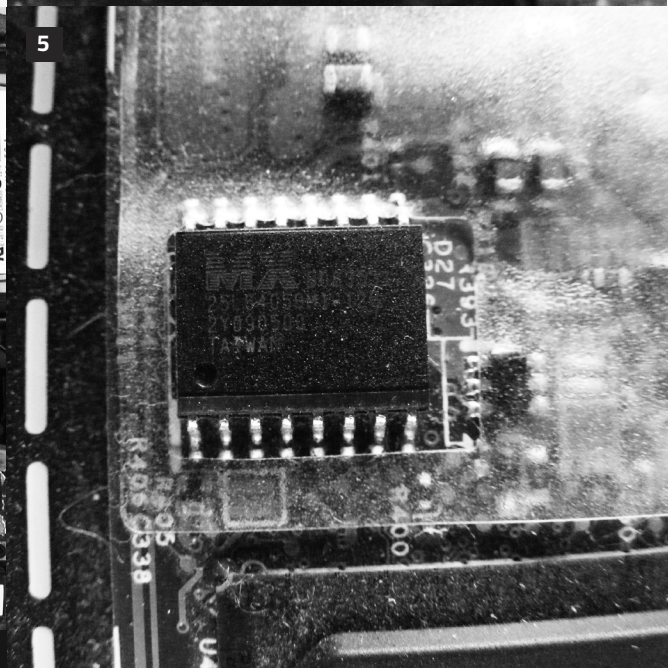
Ok, so before we get started, we'll need to make a note of the MAC address for the laptop. It looks something like this, `0A:0B:0C:0D:0E:0F`, and is a mixture of letters and numbers. You can usually find it on a sticker on the bottom of the laptop, but if it's not there, you can boot into a Linux based OS, and use the `ifconfig` command to find it.

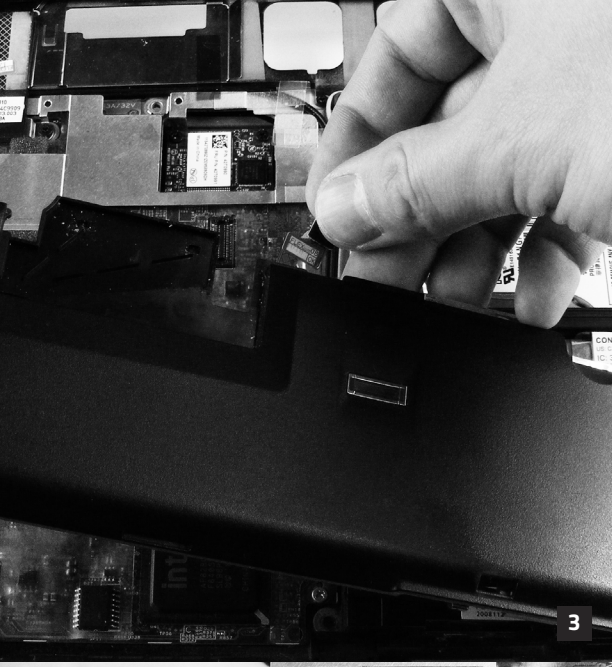

FREE AS IN FREEDOM
Search ISOLINUX menu (AHCI) [a]
Search ISOLINUX menu (USB) [u]
Search ISOLINUX menu (CD/DVD) [d]
Load test configuration (grubtest.cfg) inside of CBFS [t]
Search for GRUB2 configuration on external media [s]
Poweroff [p]
Reboot [r]
to select which entry is highlighted.
the selected OS, 'e' to edit the commands before booting or 'c' for a command-line.

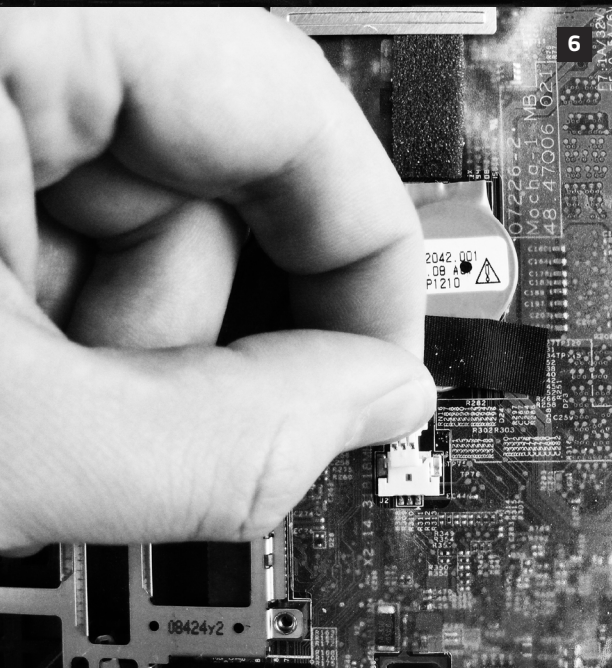

1


2


4
TOSHIBA
MK2561GSYN
LMC
K8C
intel


5





PREPARING THE X200

1. Remove the battery, then all the screws shown from the underside of the X200.

2. Place a flathead screwdriver under the front of the keyboard, and gently pry it up. Then disconnect the cable which connects it to the motherboard. Put it to one side.

3. Remove the cable to the fingerprint sensor, then lift off the palm rest

4. There, you'll see the BIOS chip. It will either be an 8 or 16 pin chip. If it has a sticker on it, remove the sticker, and use rubbing alcohol or similar to get rid of the residue.

5. We need to write down the numbers that are on the chip. This will help us identify it in a later stage. For example, mine says "MX25L6405".

6. While we're here, we can disconnect the CMOS battery too. It's the yellow coin cell above the chip.

Continued...

RASPBERRY PI SETUP

Alright, now we need to set up the Raspberry Pi so it's able to flash the BIOS firmware. I used a Pi Zero (with GPIO headers), but a Pi 2 or 3 should also work.

1. Download and burn the latest lite version of Raspbian OS to a micro SD card and insert it into the Pi. Refer to the official RPi page for a how-to (`raspberrypi.org/downloads`) if you've never done this before.

2. Hook the Pi up to a monitor, add a keyboard, and power it on.

3. Log in using the default credentials (username: `pi`, password: `raspberry`).

4. In the console, type `sudo raspi-config` and press Enter.

5. Go into Interfacing Options and enable SSH, SPI and I2C.

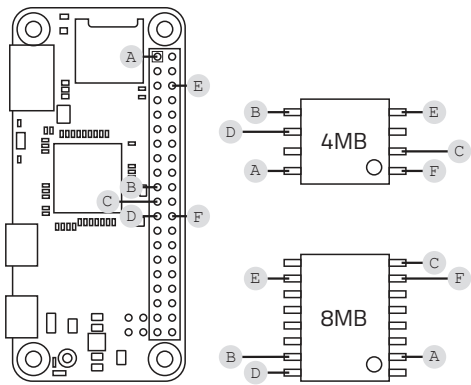
6. If you haven't already, connect the Pi to WiFi or ethernet so it's connected to the Internet. Once done, find out the Pi's IP address and and make a note of it. Enter `ifconfig` into the console and look for an interface with a local IP address. It will usually be something like `192.168.0.10`.

7. Now, you can do the rest by SSHing into the Pi from your main computer, so you can disconnect the monitor/keyboard, etc. Open a terminal on your computer and type in: `ssh pi@192.168.0.10` (replacing the IP with the one you wrote down). Type in the password to log in.

8. Let's install what we need. Type in (all on the same line):

```
sudo apt-get update && sudo apt-get
install libftdi1 libftdi-dev
libusb-dev libpci-dev subversion
flashrom
```

9. Once everything is installed, turn the Pi off, and wire up the SOIC-8 or SOIC-16 clip adapter, depending on which one your X200 has. Use the diagram on the next page for reference. Female jumper cables are much easier to work with, but you can just solder the wires directly if you don’t have any.



GETTING THE X200 FILES

First, go to `libreboot.org`, and click on Download. Click on one of the HTTPS mirror links, then click Stable. Now click on the '20160907' link.

Download the util file:

```
libreboot_r20160907_util.tar.xz
```

OK, now we need to find the correct rom files. Click the 'rom' directory, then navigate to the 'grub' directory.

Now scroll all the way down to the bottom of the page, and download one of the following files. (the 4MB file if you have the 8 pin chip, and 8MB for the 16 pin)

```
libreboot_r20160907_grub_x200_4mb.tar.xz
libreboot_r20160907_grub_x200_8mb.tar.xz
```

It's up to you how you do this, but get those newly downloaded files onto the Pi, and place them in the home folder.

(Note: I've also included them in the zine dat folder, in case the official site is unavailable in the future.)

FLASHING THE BIOS CHIP

Now we need to generate a custom ROM file to go on our BIOS chip. Turn on your Pi again, and SSH into it from your other computer.

1. Extract the Libreboot rom tarball we downloaded in the last section (all on the same line):

```
tar xf
libreboot_r20160907_grub_x200_8mb.tar.xz
```

Then do the same for the Libreboot util file (again on the same line):

```
tar xf
libreboot_r20160907_util.tar.xz
```

2. Now, navigate to the new util directory:

```
cd libreboot_r20160907_util
```

3. Go into the 'ich9deblob' directory:

```
cd ich9deblob
```

4. Navigate to the 'armv7l' directory:

```
cd armv7l
```

5. Now run this command, replacing `00:00:00:00:00:00` with the MAC address you wrote down at the start:

```
./ich9gen --macaddress 00:00:00:00:00:00
```

This creates a bunch of new files. You will need either the `ich9fdgbe_8m.bin` or `ich9fdgbe_4m.bin` file depending on the BIOS chip you have.

6. Let's create a new 'x200' directory, and copy the bin file into it. Remember to change the number depending on your X200 BIOS chip:

```
mkdir ~/x200
cp ich9fdgbe_8m.bin ~/x200/
```

7. Now we need to grab the .rom file which corresponds to the keyboard layout on your particular X200. Again, remember to change the '8mb' to '4mb', depending on your chip. Go to the rom directory, then list the files.

```
cd ~/libreboot_r20160907_grub_x200_8mb
ls
```

8. Copy the particular rom file you need to the 'x200' directory we made earlier. I'm using a UK keyboard layout, so I chose the `x200_8mb_ukqwerty_vesafb.rom` file. Type the following (on the same line):

```
cp x200_8mb_ukqwerty_vesafb.rom
~/x200/
```

9. Now there should be 2 files in our 'x200' directory. We now need to write the generated .bin file into the .rom to create our new rom

specific to our X200 laptop, so first go back to the 'x200' directory:

```
cd ~/x200
```

Then run the following (all on one line):

```
dd if=ich9fdgbe_8m.bin
of=x200_8mb_ukqwerty_vesafb.rom
bs=1 count=12k conv=notrunc
```

10. The *vesafb.rom file is now updated and tailored to your X200 laptop.

11. Connect the SOIC clip to your BIOS chip. Make sure it is the correct way around, and attached firmly. We need to back up the existing data on the chip, but first we have to find out the specific name flashrom gives your chip. Run the following (again, all on one line):

```
flashrom -p
linux_spi:dev=/dev/spidev0.0,spispeed=128
-r factory1.rom -V
```

This will output a bunch of text talking about multiple flash chip definitions, and that you need to specify the exact chip:

```
Multiple flash chip definitions
match the detected chip(s):
"MX25L6405(D)",
"MX25L6406E/MX25L6436E",
"MX25L6445E/MX25L6473E"
Please specify which chip
definition to use with the -c
<chipname> option.
```

12. Find the chip name closest to the one you wrote down earlier, and repeat the process, now with your specific chip number. (Replace 'MX25L6405' with your chip)

```
flashrom -p
linux_spi:dev=/dev/spidev0.0,spispeed=128
-r factory1.rom -V -c MX25L6405
```

This process takes around 10-15 minutes.

13. Once complete, we need to repeat this step two more times:

```
flashrom -p
linux_spi:dev=/dev/spidev0.0,spispeed=128
-r factory2.rom -V -c MX25L6405
```

And once more:

```
flashrom -p
linux_spi:dev=/dev/spidev0.0,spispeed=128
-r factory3.rom -V
```

14. You've now backed up the factory firmware 3 times ('factory1.rom', etc.). To make sure everything is being read correctly, we'll compare the shasum of each of the rom files.

```
sha512sum factory*.rom
```

If the characters match, we're good to go. If not, turn the Pi off, remove the SOIC clip, then try the backup steps again.

I originally had problems with my clip not being in place securely enough, so double check all the connections.

15. Now it's time to flash the chip. Run the following command, and remember to change the chip number to your specific one. Also note the .rom file specified is the one we made earlier in our 'x200' directory.

Again, keep it on one line (sorry, lack of space writing this!)

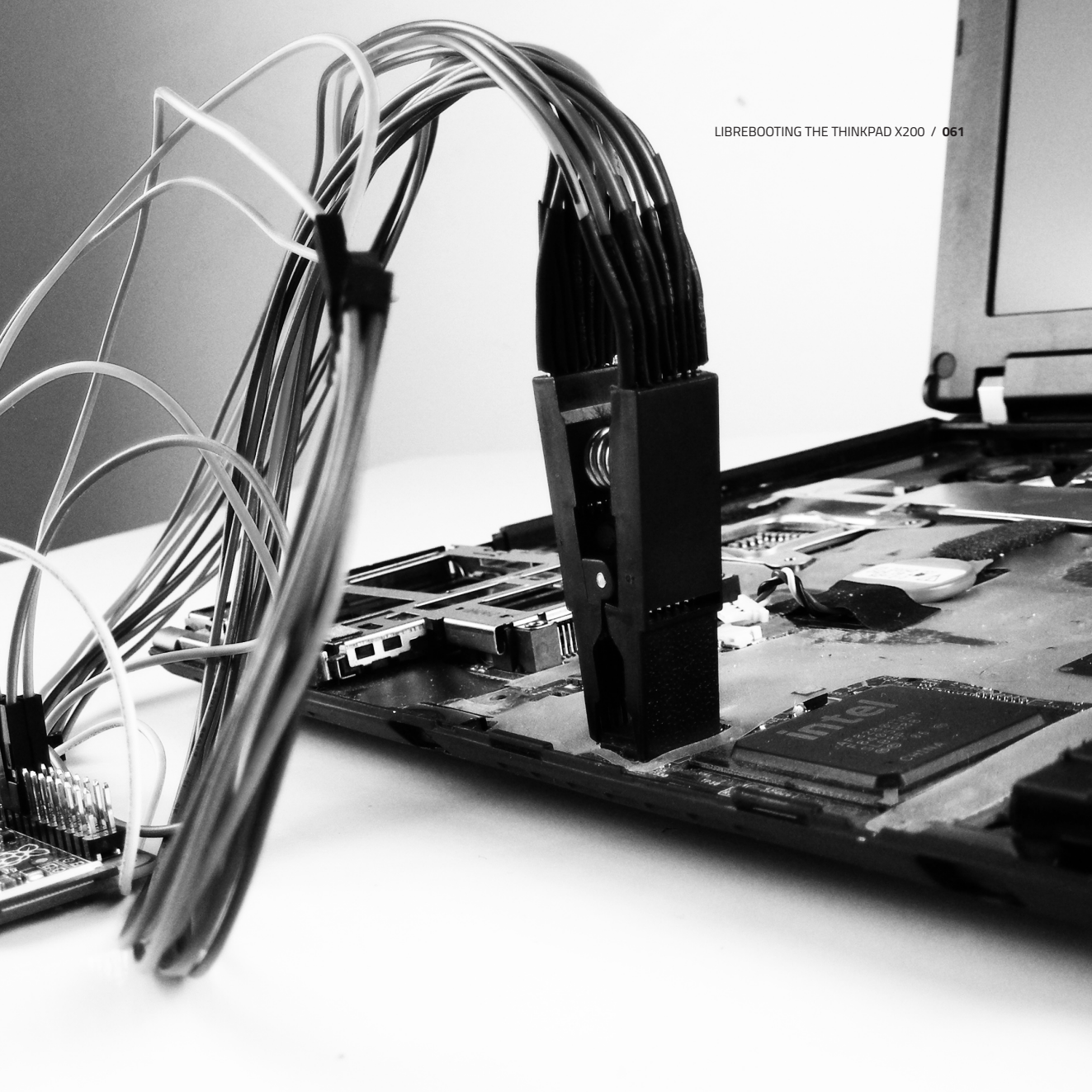
```
flashrom -p
linux_spi:dev=/dev/spidev0.0,spispeed=128
-w ~/x200/x200_8mb_ukqwerty_vesafb.rom -V
-c MX25L6405
```

This process takes about an hour. Don't be alarmed if it initially spits out errors, just keep waiting. Eventually if it says 'verifying flash...VERIFIED', the process will be completed! Make sure not to move the SOIC clip at all during this process, and if it does fail, keep trying.

16. If all's good, turn the Pi off, disconnect the SOIC clip, and power up the X200. You should see the Libreboot BIOS page. Put everything back together, and you're good to go.

You now have a laptop with dodgy boot backdoors mitigated. Next steps would obviously include installing a FOSS operating system.

This leaves you with one of the freest laptops you can own these days. For web browsing, coding, writing, light editing, etc, it's more than capable. Enjoy!


intel

DECENTRALAND: BUILDING THE NEXT VIRTUAL WORLD

Decentraland's mission is to create a social and 3D virtual world that is owned and managed by its users.

This virtual world is, at its essence, a platform including the open source tools to create dynamic 3D content, games, and applications; the infrastructure to host and distribute that content in a decentralized fashion (without relying on central servers); and a web-based app through which users can access and interact with that content.

We're excited to share our vision, our journey, and what we've learned along the way.

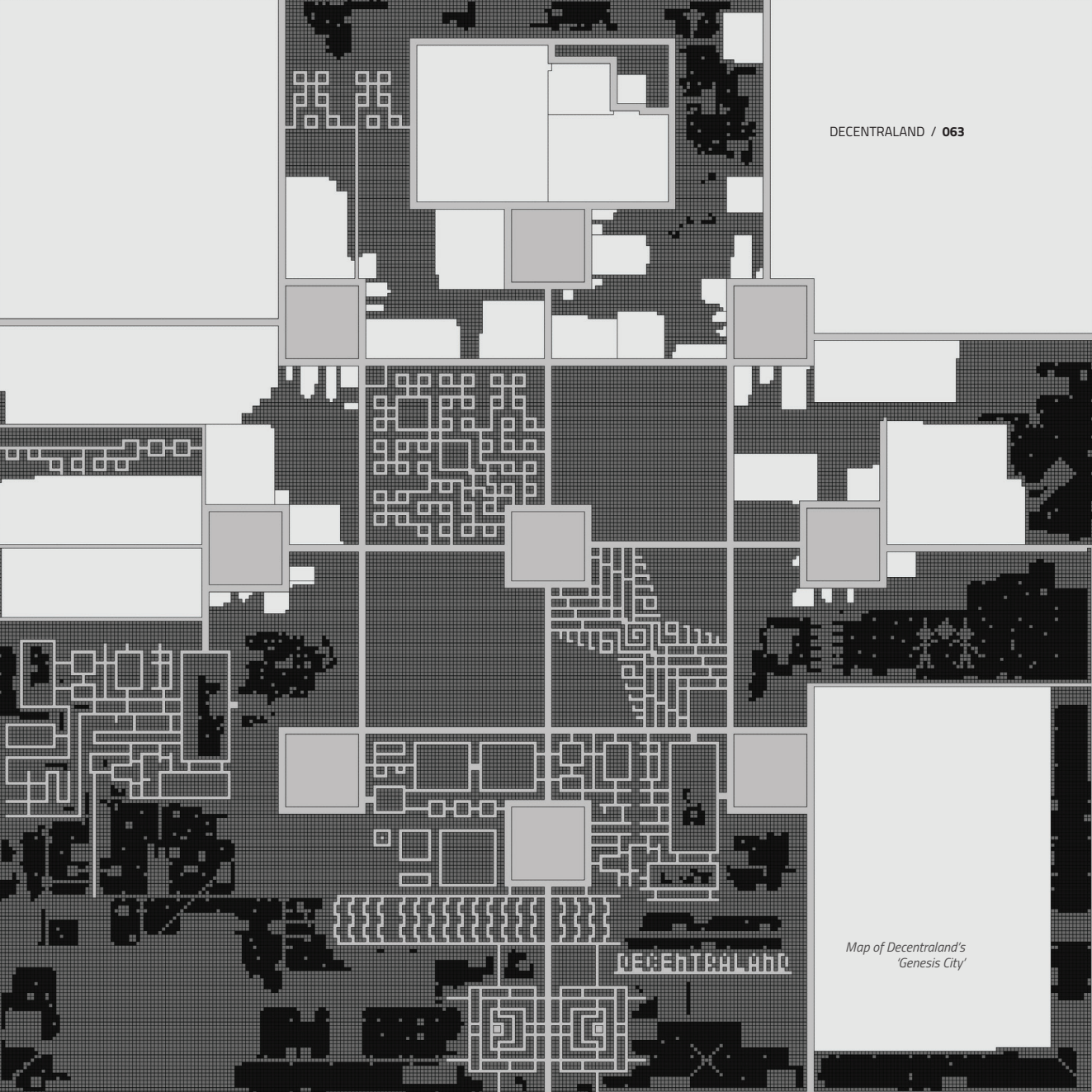
INITIAL INSPIRATION

"Science fiction is any idea that occurs in the head and doesn't exist yet, but soon will, and will change everything for everybody, and nothing will ever be the same again. As soon as you have an idea that changes some small part of the world you are writing science fiction. It is always the art of the possible, never the impossible." -- Ray Bradbury

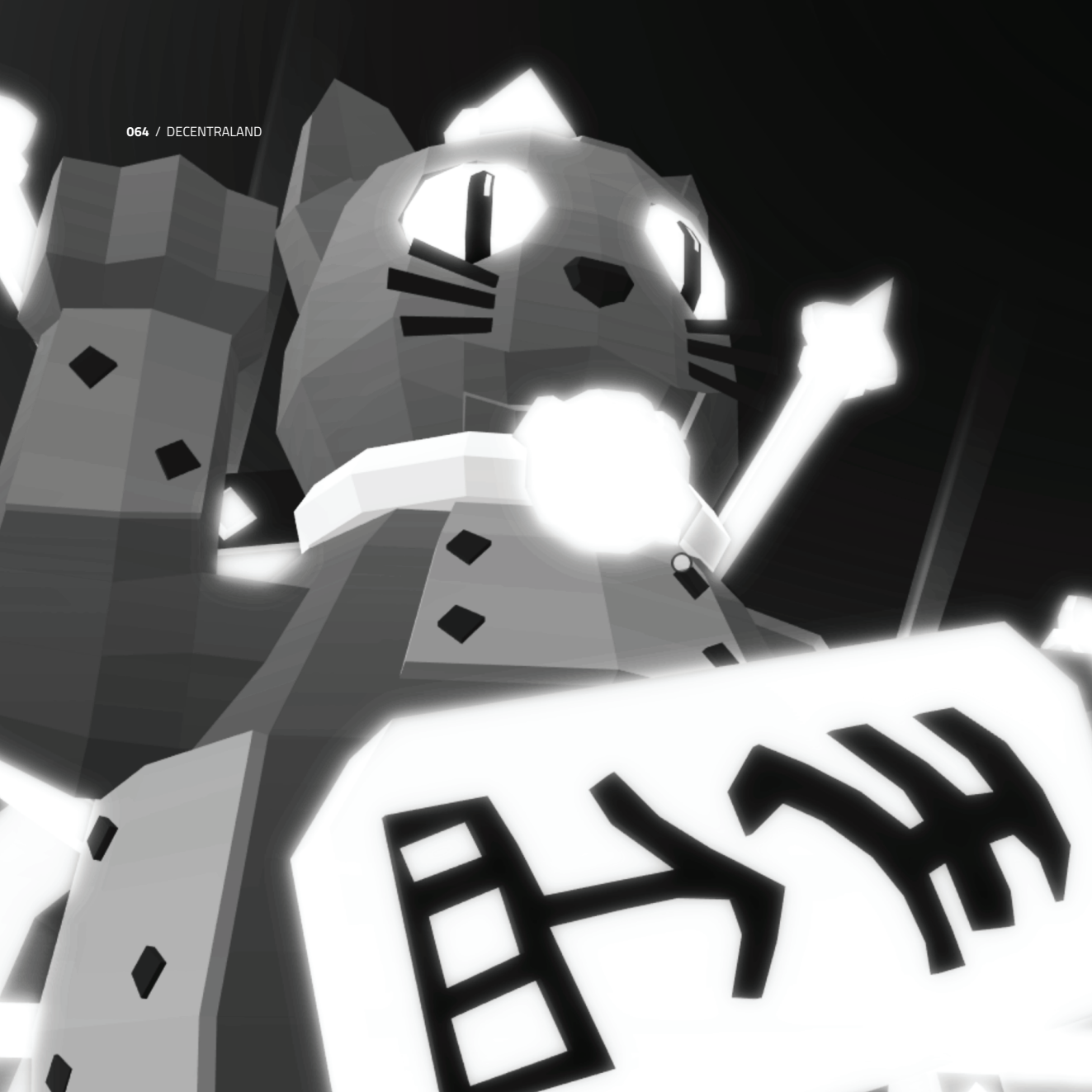
When laying down the concept for Decentraland, we definitely were inspired by various fictional virtual worlds, such as in *Ready Player One* (2011) and *Snowcrash* (1992). As an organization, our goal is to provide the foundations for a world without owners, a platform that can eventually become self-sustaining, like the web: free from the control or oversight of a central organization, with a set of open standards to which everyone can contribute.

FINDING A PATH FORWARD

We've been experimenting with different P2P architectures needed for a truly decentralized platform, all while trying to cater to very different use cases. On one side, the network has to be resilient. We cannot let any particular node become a single point of failure. On the



Map of Decentraland's 'Genesis City'



other side, we want to enable hundreds of users to connect at the same time.

Even given these constraints, we've arrived at a solution where LAND owners become brokers of messages sent over the network: this prevents the very problematic quadratic increase in the number of connections between clients. In a pure P2P network (where everybody connects to everybody else) one hundred interconnected clients results in roughly five thousand individual connections!

However, if each node is operated by a parcel owner (who serves as the "message broker"), then the number of connections falls to only one hundred connections for each parcel.

CURRENT SOLUTIONS

We saw IPFS (the InterPlanetary File System) as the clear solution for our unusual use case, but we quickly ran into some problems. We knew that we would have to allow members of the network (LAND owners) to have control over the particular content they want to serve, a task that IPFS doesn't lend itself to easily.

We eventually created an entirely new interface for IPFS nodes that resembles a more traditional HTTP API. This gave LAND owners the control they needed. With a better separation of concerns (what to display on each parcel and how to transport it over the P2P network), "seeders" can provide a far better service to developers building scenes (the 3D environments in Decentraland) and to the clients (or users) accessing that content.

We've been waiting to open-source this code only because of how heavily and frequently we've been iterating on the underlying architecture. Once we pass this milestone, the Decentraland community will be able to start building on some very solid ground.

CHALLENGES ALONG THE WAY

With such an ambitious vision, we weren't surprised to encounter some challenges. We've ended up making different trade-offs and compromises, but we've worked to stay true to our mission of building a distributed network, free from centralization of resources and power.

Navigating through the Ethereum ecosystem and discovering just what we can build in this space has been a phenomenal experience. We've met similar, and very inspiring, projects all working to cultivate a network of people who value our unique opportunity to redefine the information networks of today.

Virtual Reality at scale hasn't been an easy egg to crack for anyone. VR development is still very difficult in its current infancy, much as when we first started Decentraland.

We strongly believe that the high bandwidth of information that one can process (and generate) with a VR headset and hand controllers is very likely to create the best personal information processor we can build in the next decade. However, for the time being, and given the small number of VR systems installed at home, our first focus is to ship a mobile and a desktop version of Decentraland.

Since Decentraland is a project with inclusiveness at the core of its beliefs, we want the world explorer (or Client) to not only to work on VR headsets, but also on desktop computers and mobile devices. The number of people actively using VR headsets is very low compared to these platforms, so completing a true VR experience isn't our first priority.

Having said that, we do have some experimental builds that work on the Vive and Rift through WebVR!

BUILDING A TEAM

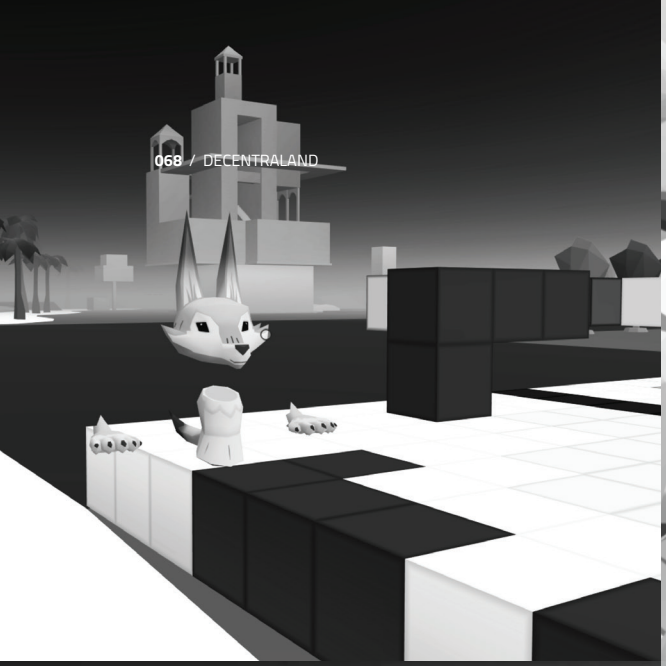
Tackling these challenges requires a small army of passionate innovators. The team at Decentraland is a global group of likeminded engineers, artists, designers, thinkers, and writers spread around the globe.

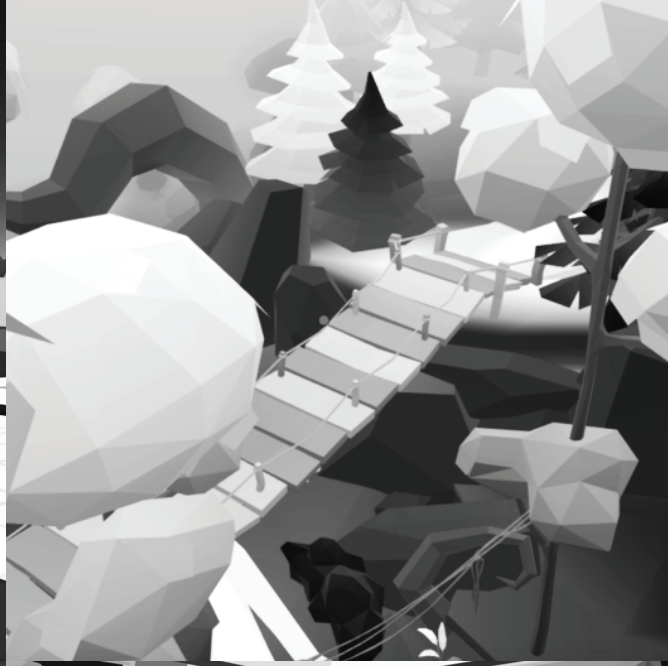
Since we first started the organization in late 2017, we've grown to include 35 members dedicated full-time to the project, mostly engineers, artists, and product designers.

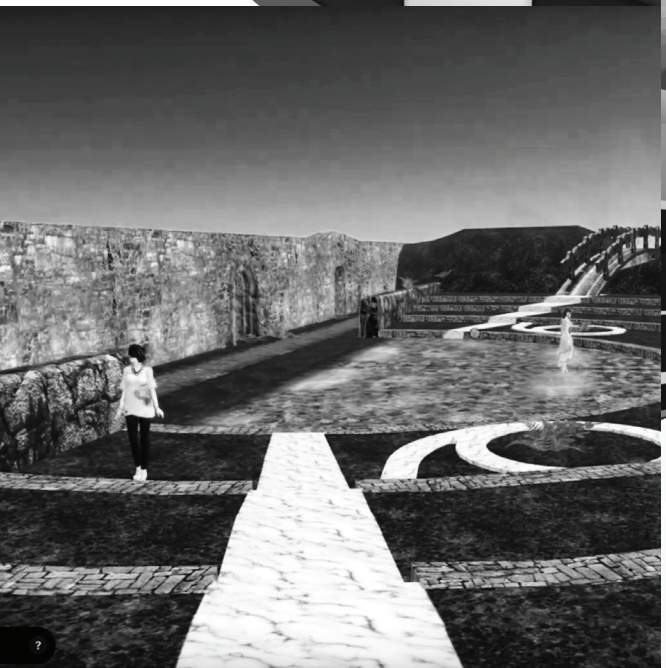
MANA: HOW WE GOT STARTED

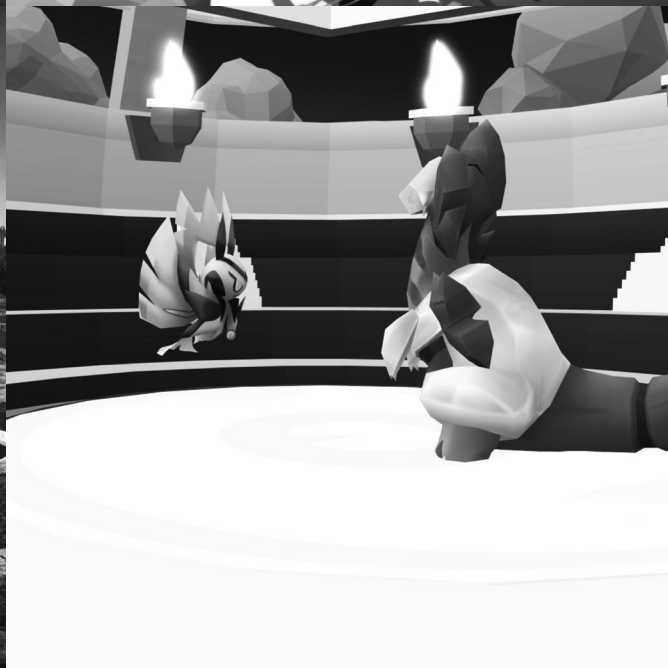
The funding for Decentraland was raised back in late 2017 through an initial coin offering (ICO) in which we issued the MANA ERC20

Screenshot from the upcoming Decentraland client









token. MANA was first used to buy LAND, the ERC721 tokens representing the parcels of virtual real estate in Decentraland, during the Terraform Event. All of the MANA exchanged for LAND in the Terraform Event was burned, removing it from circulation.

Today, MANA can used to buy and sell LAND in the Marketplace, it can be burned in exchange for LAND in our public auctions (like the second public auction we held in December of last year), and it can be used to weight votes in Agora, our community voting platform. We intend for MANA to eventually be used in many of the "in-world" transactions conducted within Decentraland.

WHERE WE'RE HEADED

We're going to release our open source client Q1 2019. Up until now, we've chosen to develop the client below the radar, to give ourselves more room to maneuver and make breaking changes.

We're preparing to release the client to a wider, more "mainstream" audience in Q2, targeting web desktop browsers, with a client re-written (but compatible) for the Unity Engine. This'll allow us to compile to WASM and benefit from a better memory management system, among many other goodies. This is going to be a major milestone for our project. By Q3, the mobile client should be released.

Once these milestones are behind us, the golden age of iterating over features and adding more value to the users of the virtual world will be our guiding star.

Finally, I want to extend our sincerest thanks to all of the district leaders, community members, and colleagues who have helped us to get here.

We'll see you in the metaverse!

Check the project out at `decentraland.org`

Written by Eric Schallock
& Esteban Ordano

(Screenshots provided by Martin Shibuya)

BEAUTY OF THE BAUD: BULLETIN BOARD SYSTEMS TODAY

In the '80s and '90s, back before the Internet became the de facto form of digital communication, Bulletin Board Systems (BBSes) were how computer enthusiasts could connect, chat, play games and share files with one another. Bulletin Board Systems were rarely run with elaborate hardware. In fact, most BBSes were simple, commodity personal computers outfitted with one or more modems and special software that could accept incoming calls from other, inquiring computers. These connections were one-to-one--each modem on the host BBS could only talk to one calling computer at a time. When a computer connected, they were greeted with an interactive, textual menu that might display colorful ANSI art and direct a caller to a chat area, a forum, a games section, or even a files directory with downloads. If all the lines on the system were busy, you'd have to wait until someone logged off before you successfully connected and got your turn. If the BBS System Operator, the SysOp, only owned this one computer, the machine could go offline if the SysOp needed to use it for another task. If there was a hardware failure, the machine may never have come back up.

Most bulletin board systems you would dial into were local to you (to avoid long distance charges), but some BBSes were famous around the world. Cities could have hundreds of them, diligently fostering unique communities while they tried to make names for themselves. Many systems had their own distinct goals, offerings, or general feel, akin to the diversity of websites we see today. Bulletin Board Systems were revered for their ability to spread information, not just through message forums, but through an abundance of text files (or "philes") that were traded from board to board. Sure, there may have been pirated warez like applications and games, but small text files written by hacking/phreaking groups, anarchists, crackers, and general computer enthusiasts went worldwide. You and your friends could write a zine full of interesting information and upload it to a local BBS. Then, overnight, people would download and

Amiga 3000 Desktop System, running a 2 line BBS, 1994
Photo by 'Acp~commonswiki' (Creative Commons 3.0)

distribute it to other BBSes all over the world. There really wasn't anything quite like it before, and some may argue there hasn't been anything like it since.

These days, BBSes have mostly died off. Those that survive are often accessible over the Internet using protocols like Telnet or SSH--something that your modern computer probably supports out of the box. A few BBSes still tie into the phone lines and are accessible via modem, but those are few and far between. Why would anyone still want to use or run a BBS these days? While retro-computing hobbyists make up a large group of modern BBS users, bulletin board systems can still be useful in specific situations. For example, consider a warring country with its Internet shut off, or a remote area where bandwidth is limited. Maybe there is a community that has no telephone access, and decides to communicate through a BBS via packet radio. Many low-bandwidth environments, including amateur mesh networks or darknets like Tor, could benefit from bulletin board systems as a lightweight way to distribute data.
Aside from that, they are a fascinating if obscure way to communicate over the wire.

ACTIVE BULLETIN BOARD SYSTEMS

There are many BBSes out there that you can still access today. Some stick around, but most live, die, and fade away in a short amount of time. Here's a list of a few popular modern Bulletin Board Systems you can check out.

Please remember, Telnet is a plaintext protocol, meaning any eavesdroppers can see your traffic. Most BBSes allow you to visit with a guest account, but if you decide to register an account of your own, use a unique password!

PIRANHA: UNDER THE BLACK FLAG
telnet - `blackflag.acid.org`

This BBS is run by the ANSI artscene group ACiD, and showcases a lot of early computer

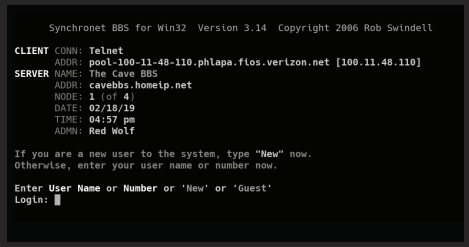
graphics artwork. Aside from the impressive interface, Black Flag also connects to the FidoNet network, a store-and-forward system for inter-BBS communication that allows for the exchange of forum messages and email. Black Flag has been operating since 1995 and is located in Hudson, FL, USA.

LEVEL 29

telnet - `bbs.fozztexx.com`
phone - `916.965.1701`

Level 29 is a popular BBS, drawing in many retro computer owners who attempt to connect from a variety of older hardware.

Unlike most currently-running BBSes, Level 29 runs its own custom software, and features an old-school dial-up number if you want to connect over phone lines. The Level 29 BBS is located in Fair Oaks, CA, USA.



THE CAVE

telnet - `cavebbs.homeip.net`

The Cave is a Windows-based BBS running the popular SynchroNet BBS software. It features a live chat system and many classic DOS door games such as Trade Wars 2002 and Legend of the Red Dragon (LoRD).

Originally operating between 1992 and 1998 in Raleigh, NC, USA, The Cave was completely rebuilt in 2004.

Continued...

ALCOHOLIDAY

telnet - `alco.bbs.io`

Alcoholiday is a BBS featuring a lot of great artwork, but it isn't your traditional art-focused BBS. Instead, Alcoholiday focuses on messaging and user communication, connecting up to 12 different store and forward networks!

Originally established in 1995, Alcoholiday was revived in 2001, and again in 2015 where it currently runs off of a Raspberry Pi in Columbia, SC, USA.

CONCLUSION

Additional BBSes are actively indexed at the Telnet BBS Guide, an online directory of bulletin board systems, at `telnetbbsguide.com`. As of this writing, there are 451 Bulletin Board Systems listed on the site with 38 new ones listed in the last 30 days alone! There are many to explore, far exceeding what's been showcased here.

If you are itching to run your own BBS, there is plenty of software available to set one up such as the classic Mystic (`mysticbbs.com`) or x84 (`github.com/jquast/x84`).

Bulletin Board System software has a relatively small footprint, and can run comfortably on small, low-power, single-board computers like the Raspberry Pi.

Have an old or underpowered computer laying around? You might have the perfect BBS machine right under your nose!

Written by Mike Dank

BBS on a Raspberry Pi

MAINTAINING PUBLIC SPACES IN P2P NETWORKS

Suppose you create something for other people to use; let's say some sort of a personal transport. *'Electric rollerblades.'* So, you have all these cool kids whizzing around in your thing. It's a new way of transport, it's green, it's useful, and it helps with congestion in cities.

Then one day, a bank robbery happens. It turns out the thieves brandished their electric rollerblades prominently, and their escape was aided by the fact that your rollerblades were able to zip around and move in places that no police car could possibly go.

Two weeks later, another robbery happens. Also on rollerblades.

Three weeks on, a woman is assaulted by someone trying to take her bag and whiz away. On the front page of *New York Times*, there's a photo of a badly injured woman in a neck brace, and beside that, a photo of a person, arrested, wearing your very own product. *Buzzfeed* says maybe it's time to ban electric personal transport on city grounds.

This ain't good. Because of the actions of a few poorly-socialised brutes, your invention, which *legitimately improves the lives of the vast majority of regular people that use it*, is in danger of being nuked from orbit. Not only in the books of law, but also in the eyes of people. It's cementing itself as a tool for crime the more they see it in the news.

This is a good moment to stop and think about why this is happening.

Your thing is *new*. It is not a car--no one thinks of cars primarily as vehicles for crime. People have little experience with your thing, however, so their impression is vulnerable to a few bad incidents on the news. There is no established precedent. Consider how smoking is a known danger, yet allowed on *personal liberty* grounds, but e-cigarettes get banned left and right.

We want our work to reach as many people as possible to help improve lives. But, even with

our best intentions, we ended up in a place where the tech is tarnished to the point that no respectable person would ever touch it.

This is a *contrived example*. But the point is, while in theory the technology stands on its own, in practice, new technologies aren't just judged by their merits, but also by how they interact with the rest of the society.

P2P PARALLELS

This has a decent few parallels with that new P2P project that you might be thinking about starting: likely it involves people; likely it involves people interacting with each other.

The point is, even if you can design your system in a way that it does not give you any power of decision-making, the very act of designing the system in the first place, is the *ultimate* kind of decision-making that you can never really waive.

You might as well get better at making these kinds of calls, because any halfway successful project is going to have to contend with this.

This isn't a theoretical problem either: this is something that I faced with Aether, and this is the thought framework that I came up with to handle these kinds of issues. To make for a better, real-world example, here's an actual issue and how I ended up with a way out.

THE GREATER GOAL

Aether is aimed at the mainstream: the goal of the project is to improve the privacy of the average user while using mass communication, and the best way to improve the 'privacy quotient' of the whole planet is to reach out to as many people as possible. This is the ultimate goal, and the success of the potential solution is judged by this.

PROBLEM

For some unknown reason, on P2P networks, people tend to post a lot of porn. That makes it impossible to have a mainstream appeal; there is nothing wrong with porn, but no one wants to open up an app downloaded from the Internet, at 3 p.m. the afternoon, at work.

Attempt 1. Block Porn!

What happened? Not only did it not work, technically, it's also that the whole thing about P2P networks is to give people more free choice, not less. In other words, you've burned the house to get rid of the bed bugs. Yes, you've achieved your first goal (via magical tech that doesn't exist, somehow), but then you've killed the whole product to be able to do so. Not cool.

Attempt 2. Mark NSFW content, and remove it from the front page.

Heil dictator! You just inadvertently installed yourself as an arbiter of what is safe and what is unsafe. With your new job as *The Ultimate Arbiter Of Good Taste*, you can expect to be doxxed, if you're lucky, every other month at most. Ouch. This is a bad idea for other reasons, such as you having to review a lot of content every day just to keep up with it.

Attempt 3. Implement a SFW list, having it enabled by default, also with the option for users to disable it.

Eh. Some people are complaining that this is effectively censorship because they can't immediately see NSFW content without flipping a switch, and that disabling the SFW list is relatively hard for beginners (you have to edit a config file for this one).

This is getting better though. You're still receiving complaints, but they're now starting to be more about what you're trying to accomplish (a *friendly*, safe-for-work front-page for beginners) and not about you being a tyrannical dictator.

Attempt 4. Keep the SFW list, but let users know when NSFW content is there, and have them click if they want to view it.

This is getting better, but also more complex. How do you know that? Because now you're getting people confused about what the SFW list actually does. They seem to think it also applies to threads, posts, or people, instead of only communities--they believe it is now more granular than it actually is.

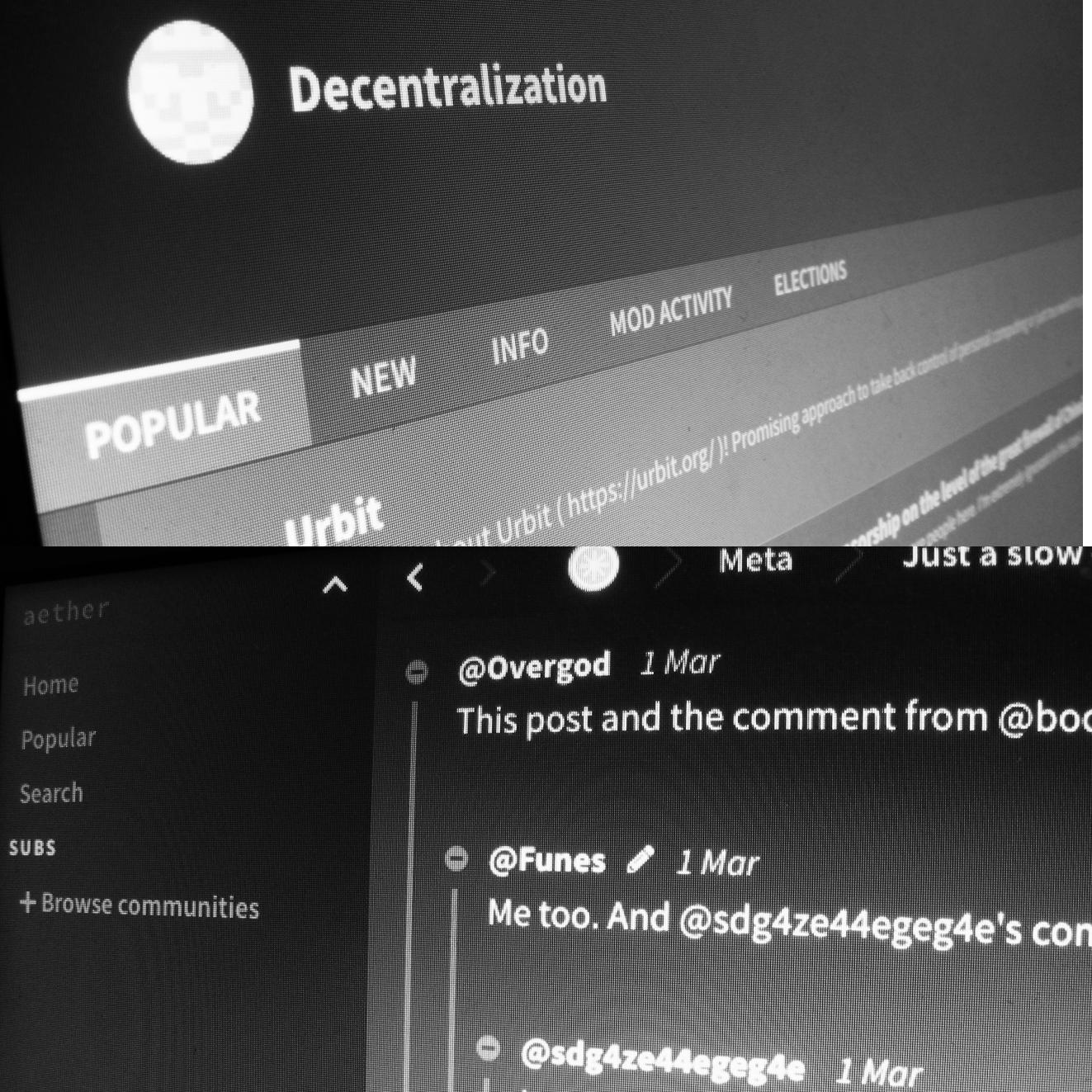
Explaining takes too long, and that alone is a sign you have to simplify the mental model needed to understand it. Also, you haven't fixed the garden variety dictator issue: you are still the only person in control of that SFW list.

While the list is optional, and further, temporarily overridable even when enabled, it does help with not binding people by your personal wishes. It does not, however, fix the *single point of decision* issue.

Attempt 5. Keep everything from previous attempts, but now introduce custom SFW and block lists that users can make and share.

Now this makes a lot more sense--you have a 'feature' that you can plan and build around, and you're no longer building a specific, exception case. That's good. People still complain, but they now complain about the actual thing you want to do (have a beginner friendly homepage), and not the things you don't want (becoming a dictator via deciding on what can be said).

This can go on for a sixth attempt, but I want to stop here to make obvious the basic pattern we are following. The main purpose of what we're doing with these iterations is to allows us, the builder, control over the amount of complexity we are willing to trade for a better solution. From the most naive to the most complex, this makes more visible what amount of complexity you're adding in each


Decentralization
POPULAR
NEW
INFO
MOD ACTIVITY
ELECTIONS
Urbit
Meta
aether
Home
Popular
Search
SUBS
+ Browse communities
@Overgod 1 Mar
This post and the comment from @boo
@Funes 1 Mar
Me too. And @sdg4ze44egeg4e's com
@sdg4ze44egeg4e 1 Mar

step, and to what improvement. Not all problems *need* complex solutions, but all problems can have them. It is up to you, as the maker, to determine how complex, thus surgical, of a solution you're willing to build for that particular problem. You will find that many problems can be solved to 80% by something very simple, and you'll feel personally responsible for solving the others to 99.8%. That is up to you. Just pick your complexity battles carefully, because I can almost guarantee that there is *some* part of your application that would appreciate your attention more.

CONCLUSION

I hope this ends up being a useful method that you can utilise when you're building your own project, as well as being informative for users, to explain the thought-processes behind making these kind of systems.

For this method, it's not a be-all-end-all way of thinking but I've found it helps more often than not. By applying this concept to a problem that I've had, I've ended up with a couple iterations on the app that (*I believe*) improved things.

For the problem in question, I'm afraid I have no great answers other than a solution being dependent on the type of community you're interested in building. Some communities below a certain size might not need this kind of design method. An example of this would be if you're creating an app that handles private communities like Discord, or Slack, where each community would have their own reputation, and it wouldn't affect the app itself. But, if you're building something like Mastodon, then most things in large communities are going to affect how your app is perceived by the public, fairly or unfairly. This happens whether you accept it or not, so the practical way of handling it is to be aware of this happening and be conscious about it. You might decide to not care, and that is fine as well--so long as it is an explicit decision to do so.

Feel free to try out my project, Aether, at `getaether.net`. We're a small, friendly decentralized community.

Written by Burak Nehbit

MESH NETWORKING BASICS

Whether for fun or necessity, numerous groups around the world have been creating wireless mesh networks to provide Internet access to local communities.

Mesh networks are easy enough to grasp in concept: nodes within a network connect directly, dynamically, and non-hierarchically to as many others as possible for the purpose of routing data to/from various clients.

In contrast to a star or tree topology you might see in Local Area Network (LAN) or Internet Service Provider (ISP) infrastructure, mesh networks lack centralization, so no one node becomes a single point of failure. Meshnets are often praised for their resiliency and scalability as they self-heal when a node goes offline, and can experience bandwidth increases when new nodes are added.

Modern mesh networks are often run by technical, grassroots groups that thrive in a given geographical region. That said, it isn't too hard to bootstrap a new mesh network by yourself using off-the-shelf hardware and open-source software.

BUILDING A MESH

The core component of a mesh network is the router. Any machine capable of wirelessly moving data is suitable for the task. Some groups choose to use standard wireless routers available in retail stores from brands like TP-Link or Ubiquiti while others may runs custom, white-box Linux machines or Raspberry Pis and similar SBCs.

Router hardware within a network can be configured to only route packets between nodes, or additionally act as Wireless Access Points (WAPs) to allow client connections locally--much like how you might think of the all-in-one wireless router on your home network (which usually consists of a router, switch, and wireless access point combined).

Routers do all the heavy lifting for the network, forwarding traffic from node to node

Guifi.net Supernode Installation in Spain
Photo by Lluis tgn (Creative Commons 4.0)

Freifunk Antenna Installation, Berlin 2013
Photo by Boris Niehaus (Creative Commons 3.0)

so it can reach its final destination. Choosing router hardware mostly comes down to hardware compatibility and availability; if the hardware is closed off or scarce, it will be harder to configure and deploy.

But how does the traffic actually get from one node to another? Mesh networks use their own routing protocols to send packets across the network. These protocols might take in to account number of hops to a destination (sum total of the routers that the traffic has to pass through), signal quality, or even bandwidth constraints when constructing paths for data to follow. Further, some protocols rely on the origin router determining the whole path for the data across the network while others allow individual routers to pick the next hop themselves, so each router determines the next link within the chain. Routing protocol choice can often come down to personal preference, ease of set-up, or performance goals. Different protocols can have different trade-offs, and it's difficult to switch packages once hardware is deployed in the field. Popular protocols you might hear about include

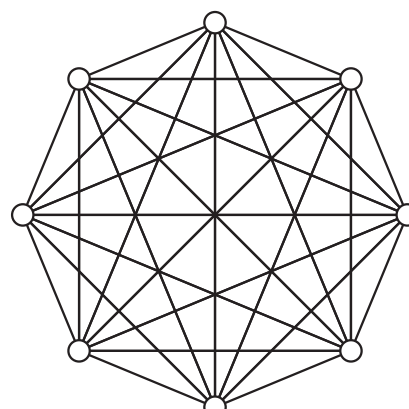
Freifunk Antenna Installation, Berlin 2013
Photo by Boris Niehaus (Creative Commons 3.0)

B.A.T.M.A.N. Advanced (commonly referred to as batman-adv), OLSR, cjdns, Babel, bmx, Yggdrasil, and others, with batman-adv being one of the most popular. Through a simple install command and a few properties set within configuration files (network name, IP address, etc.), many of these protocols can be up and running in minutes.

Routing protocol choice goes hand-in-hand with operating system software that runs on any given piece of router hardware. The most popular third-party operating system for wireless routers is OpenWRT, an open-source OS forked from the WRT software that originated with the once-popular Linksys WRT line of devices. OpenWRT can accommodate a variety of use cases, including mesh networking, and is considered much more mature than alternative packages such as DD-WRT and Merlin. Some wireless routers offer their own first-party software to perform mesh networking, but this is usually not robust enough for a wide, public deployment when compared to the use-case of a home network setup. Some hardware companies like Ubiquiti

and MikroTik have first-party software with robust mesh networking support, but bear in mind that this is proprietary and closed--geeks trying to run their own routers will have to buy specific hardware and the network will rely on the Original Equipment Manufacturer (OEM) for support and updates. Hardware that directly supports and runs a Linux variant can be incredibly flexible, but more difficult to setup and maintain. A standard distribution such as Debian can easily support a wide variety of mesh networking protocols given it has compatible wireless radio hardware.

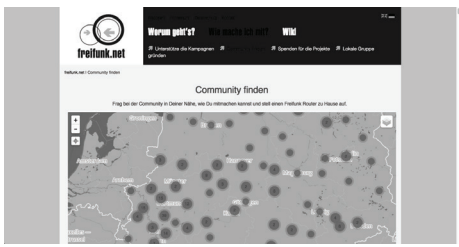
Some networks may also choose to allow Internet access for clients connecting to nodes (in addition to mesh-only sites and services), but they usually don't rely on standard residential or business connections. These networks will often set up a Point of Presence (PoP) at an Internet Exchange (IXP), which is essentially a physical location for different organizations, companies, and providers to connect with one another for network access. When two entities want to connect directly with each other, usually for the mutual benefit of easy access to the other's network, they peer with one another using an older protocol known as Border Gateway Protocol (BGP). BGP is responsible for routing Internet traffic all over the world through a series of point-to-point links between two entities and their respective networks. Taking a step back, BGP itself creates links to form a type of mesh network, and the more connections a node on the Internet has, the better.



Many of these links are done without money trading hands, and it is easy to understand how two companies like Comcast and Netflix might want to peer so customers will have better (faster) access to content they want.

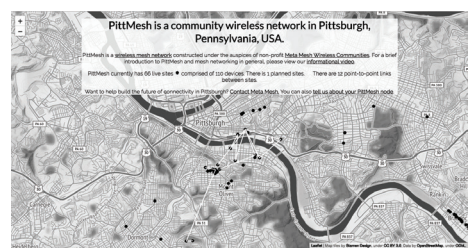
CURRENTLY OPERATING NETWORKS

So let's take a look at how some existing mesh networks operate.



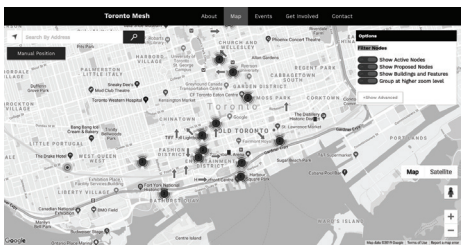
The Freifunk Project originating in Germany aims to establish free, independent and non-commercial network access for local communities. Over 400 communities in and around Germany, boasting over 41,000 access points, make up the sprawling network. Freifunk uses fairly standard off-the-shelf wireless routers for their access points, as several communities within Freifunk build and maintain their own firmware images based off of OpenWRT.

The exact firmware distribution used will depend heavily on the region, but most seem to support TP-Link and Ubiquiti devices. Much like the differences in firmware deployment, routing protocol choice can differ from region to region as well. Some communities rely on the older OLSR protocol, while others are moving to the more modern batman-adv. While batman-adv is now preferred for routing, some groups push the envelope and use more experimental protocols like bmx6/bmx7.



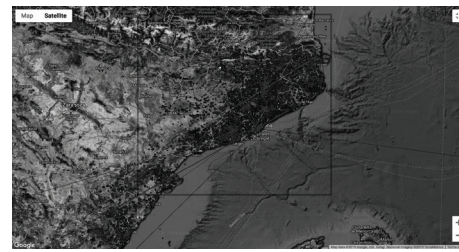
PittMesh, located in Pittsburgh, Pennsylvania, USA, is a community wireless mesh network run by the MetaMesh organization. PittMesh nodes are primarily deployed by MetaMesh, consisting mostly of GL.iNet devices to form a network of over 60 devices. GL.iNET devices are manufactured to run OpenWRT out of the box, making them easy to use with custom

software. PittMesh nodes run OLSR to route within the network, and any device can become a node provided it also runs OLSR.



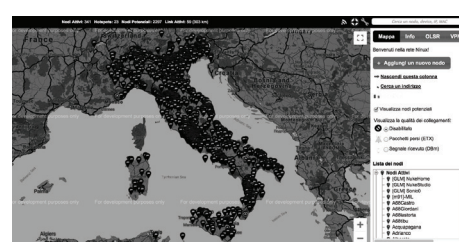
Toronto Mesh, located in the city of the same name, is a community mesh initiative working to not only build its own network, but create tools and documentation to allow others to do the same.

Nodes are primarily created with single board computers like the Raspberry Pi, Orange Pi, and ROCK64. These nodes leverage separate WiFi dongles for 802.11s networking between nodes, and cjdns software running on Linux to provide routing. Toronto Mesh only has a handful of nodes deployed, but they are constantly refining their software and testing new hardware that may suit their network.



The **Guifi.net** project in Spain aims to create a free, open, and neutral network for all to use. Over 35,000 nodes (and counting) are currently deployed on four continents.

Initially, Guifi.net used commodity hardware and their own "dd-guifi" firmware (based on DD-WRT), but now mostly use MikroTik routers running RouterOS firmware along with proprietary routing protocols.



Ninux, a mesh network project in Italy, is a free, open, experimental network. Ninux follows many principles established by other European mesh networks, and shares testing information to improve mesh networking practices as a whole. Ninux has over 340 nodes deployed which run their own FirmwareNG firmware based on OpenWRT, utilizing the OLSR routing protocol.

The network has a diverse mix of devices running as routers, but currently recommended devices are manufactured by TP-Link and Ubiquiti.

CONCLUSION

Mesh networks are a simple concept that are highly customizable and relatively easy to set up. With a small investment in commodity hardware, anyone can get a multi-node mesh network up and running in an afternoon with open-source software that is readily available.

These days, many are worried about increased government meddling and threats to net neutrality as service providers consider changing how we can access the Internet. Wireless mesh networks are a community-focused way of creating an open, neutral network, and more of them are starting up all the time. There might already be one in a city near you!

LINKS

```
https://open-mesh.org
https://openwrt.org
https://olsr.org
https://github.com/cjdelisle/cjdns
https://freifunk.net
https://pittmesh.net
https://guifi.net
https://tomesh.net
http://ninux.org
```

Written by Mike Dank

3D PRINTING WIFI ANTENNAS

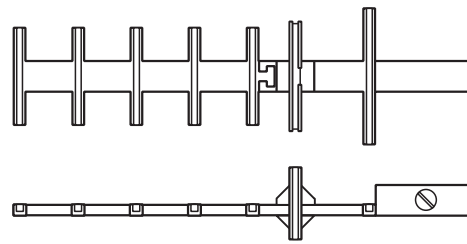
This guide will show you how to create two different long range Wifi antennas, using cheap, readily available parts, and a 3D printer. Both antennas cost a few dollars each to make, and have the potential to reach maybe 2 miles when pointed accurately with good line-of-sight.

I've seen similar instructions online, but they involved a lot of manual measurements and assembly. Making the antennas 3D-printable removes a lot of those steps, and makes the process more streamlined and precise.

(All files needed for these projects are in the dat mentioned at the beginning of the zine)

MINI YAGI ANTENNA

This little yagi antenna is built with size in mind, and can be assembled/disassembled easily by simply clicking everything together.



PARTS

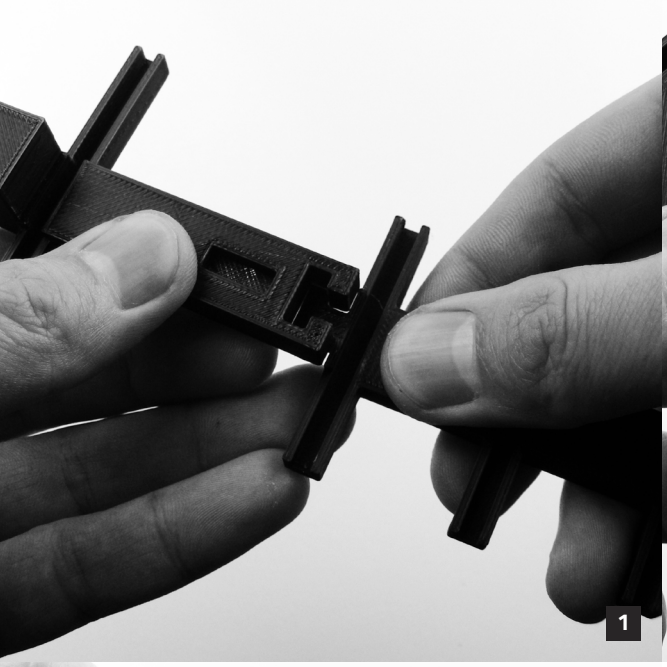
- 3D Printed Antenna Parts
- USB Wifi Dongle with SMA adapter
- 14AWG Flexible Silicone Wire (~1m length)
- Male to Female Reverse SMA Cable
- M6 x 6mm Screw
- M6 x 10mm Hex Standoff
- Small rubber band

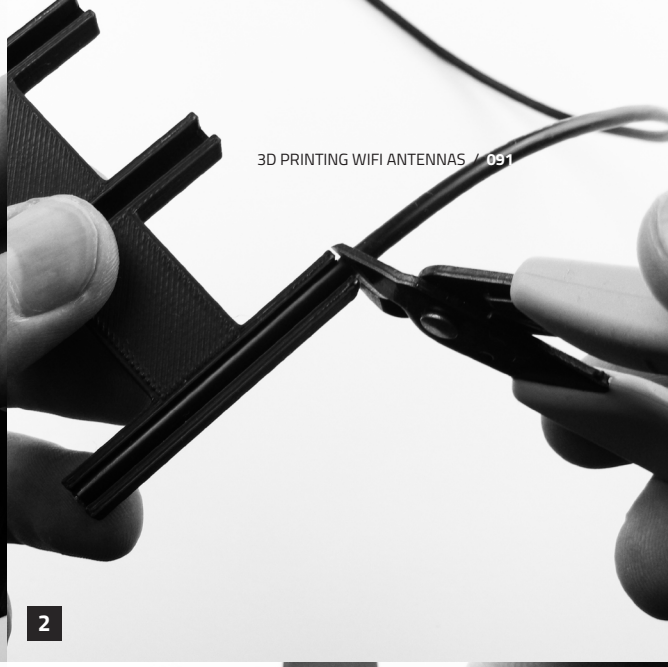
ASSEMBLY

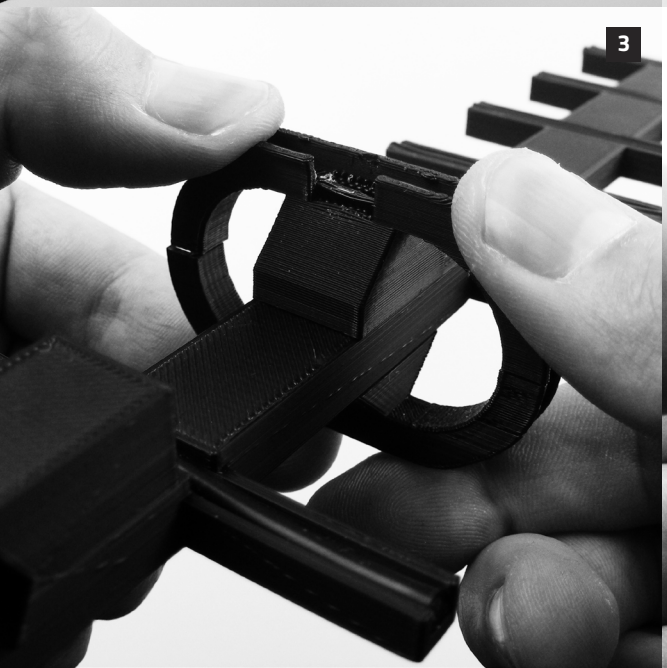
1. 3D print all the antenna pieces, and snap the two halves together.

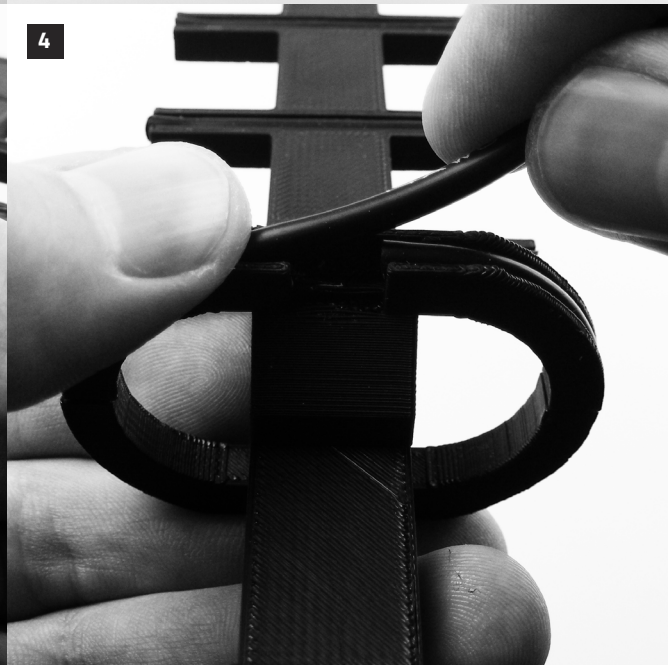
2. Insert the wire into the spaces (they should fit snuggly), and cut the ends so they're flush.

3. Push the two halves of the driven element


1


2


3


4

into the antenna. Make sure the top half has the opening facing toward the back of the antenna.

4. Wind the wire around the driven element, making sure it starts at the opening, and ends a little after the end of the opposite side.

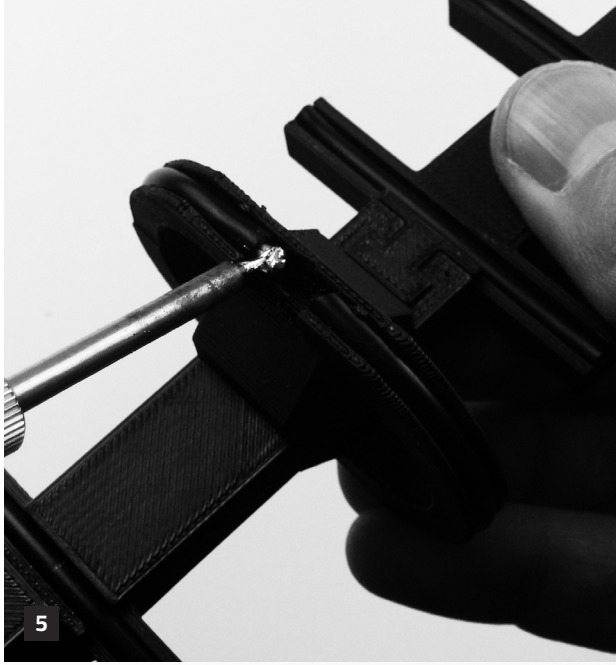
5. Cut the wire, strip 5mm off the end, and tin the end with solder.

6. Take the female end of the reverse SMA cable, cut it to about 5cm, and strip the end so the inner core is exposed. (Make sure to cut the outer shielding away). Then solder the SMA cable to the wire on the driven element.

That's basically it! You just screw in the tripod mount, connect your USB wifi dongle, and you're good to go.

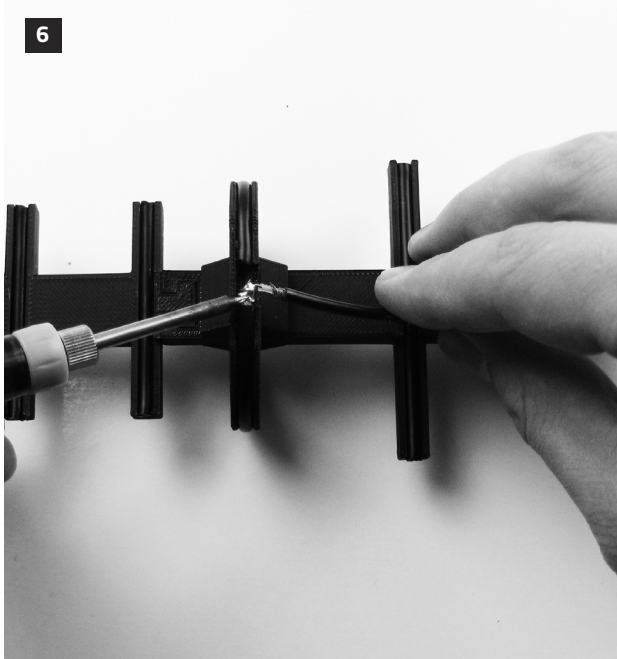
This yagi design is pretty decent as a general antenna, and does a good job at both boosting signals, and finding more access points. Not bad, consdering it's super low cost.

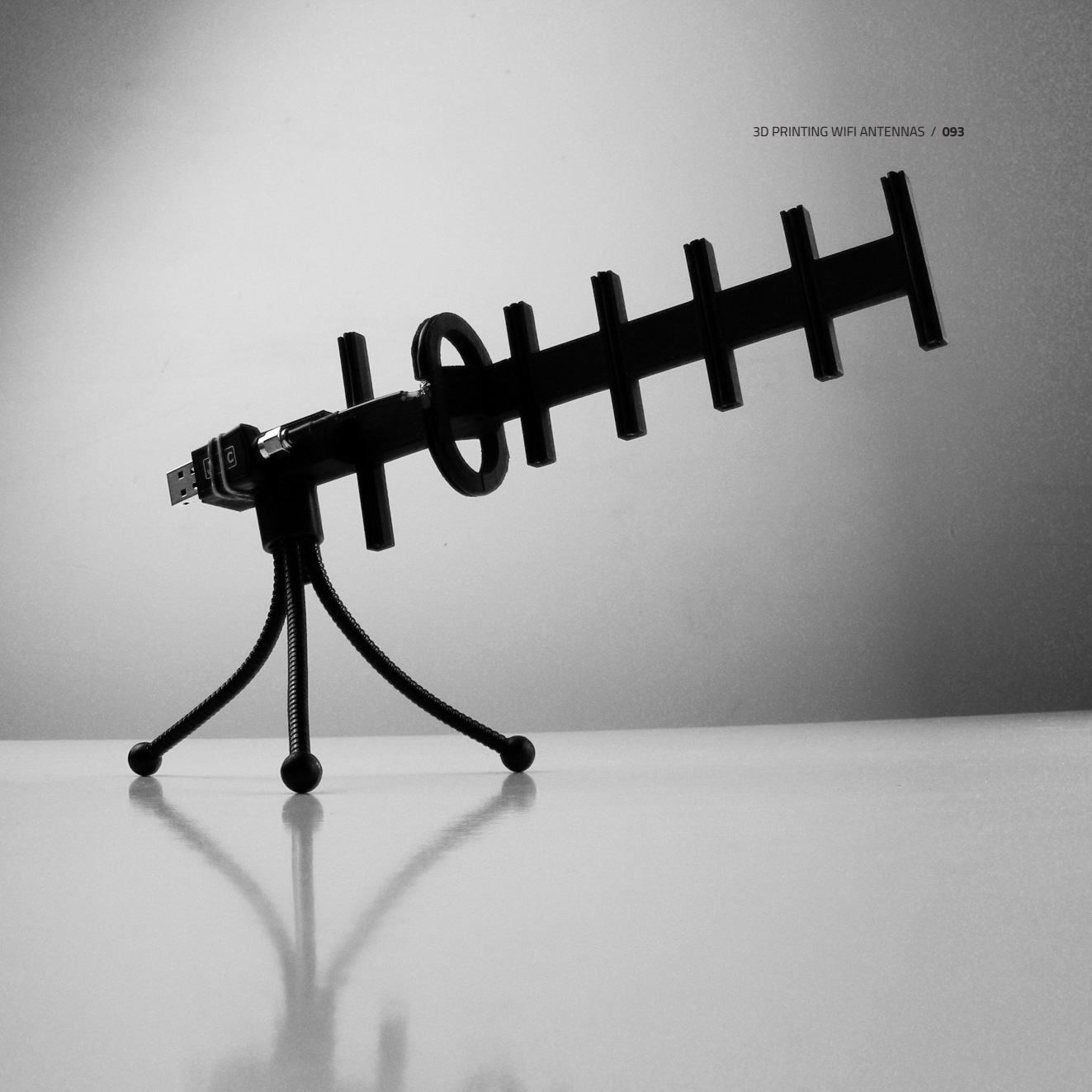
Turn over to see the helical antenna design.



5

6





HELIX ANTENNA

This design is bigger, and works better outdoors, so it should be good for meshnets and the like. It's kind of like a more polished Pringles "cantenna," in that it uses a helical design to catch the WiFi waves.

(This design is an iteration of the one documented at `ab9il.net/wlan-projects` so check that out for more advanced details on the construction.)

PARTS

This design uses many of the same cheap, and easy to find parts as the yagi antenna.

- 3D-Printed Antenna Parts
- 14AWG Flexible Silicone Wire (~2.5m length)
- USB Wifi Dongle with SMA adapter
- Male to Female Reverse SMA Cable
- M6 x 6mm Screw
- M6 x 10mm Hex Standoff
- 4x M2.5 x 15mm Screws
- 110x110mm Groundplane PCB (or standard single-sided copper clad board)

ASSEMBLY

1. Print out all the parts, and assemble the tube sections. The three parts are numbered, with "1" going at the bottom. Make sure all the eyelets are lined up.

2. Now we need to wind the 14AWG wire around the antenna tube. Make sure that the tubes remain aligned, and make sure there is enough excess wire at the bottom.

3. Take the PCB, and with the NODE logo at the back, align it up with the antenna tube. Like the other antenna, you'll need to cut and strip about 5mm off the end of the wire, but the exposed part needs to line up with the PCB. (As shown)

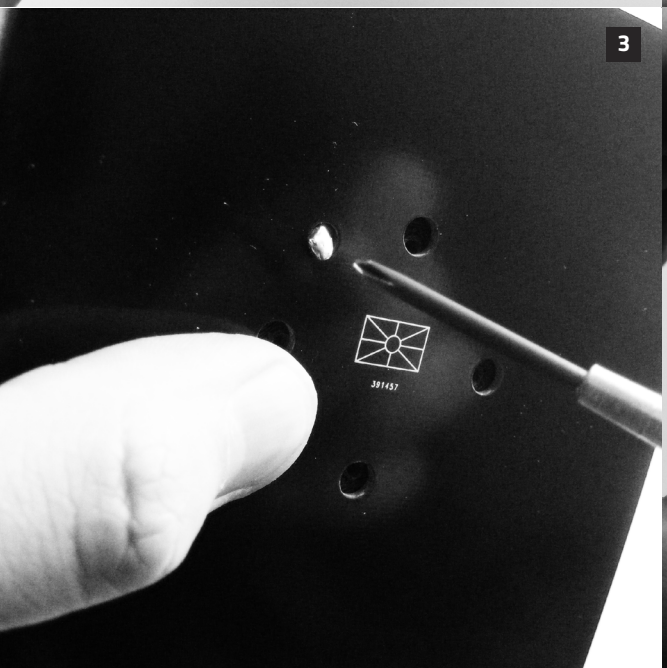
4. Take the female end of the reverse SMA cable, cut it to about 5cm, and strip the end so the inner core is exposed. (Make sure to cut the outer shielding away)

5. Solder the 14AWG wire to the SMA cable, through the hole in the back of the PCB.

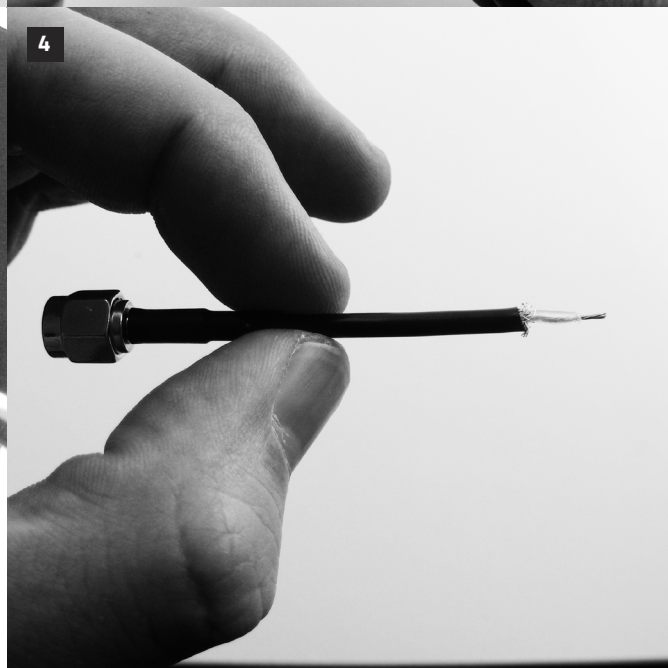
6. Once complete, use the 4x15mm, and screw

1

2



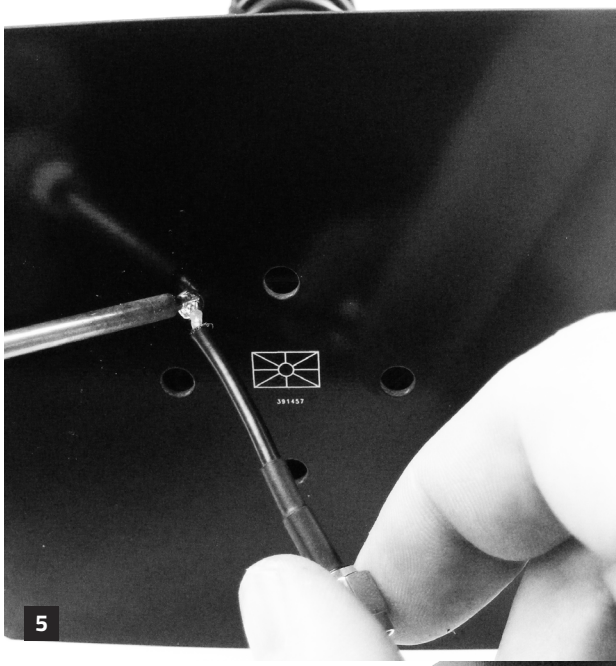
3



4

the mount through the back of the PCB and into the bottom of the tube base. If you have a M2.5 thread tap, this would make it much easier. Add the tripod mount screw and hex nut, and you can now connect up the USB dongle, and start using it.

I connected mine to a larger tripod for accurate directionality, but you could also make some sort of bracket for the back if you wanted to screw it directly into a wall. If you're using it outside, also remember to waterproof the SMA connector and USB dongle somehow.



5

6

CONCLUSION

So those are the 3D-printable yagi and helix antennas. Like I said, both of these should reach a mile or two, at 2.4GHz, depending on a clear line of sight. The helix design is a little more temperamental, and needs to be pointed exactly in the correct position, but apparently they work much better outside, and through environmental factors such as rain & snow.

If any antenna experts have ideas on how to improve these, email `mail@n-o-d-e.net`.

THE RISE OF SMART ORGANIZATIONS

In 2009, Satoshi Nakamoto mined the bitcoin genesis block, setting off a revolution in the interdisciplinary fields of cryptocurrency and smart contract design. Initially marketed as a "peer-to-peer electronic cash system," bitcoin and the blockchain data structure that secures its transactions have gone on to be used for applications as diverse as crowdfunding, document time-stamping, and virtual reality. With the advent of decentralized digital currency controlled by rules encoded in smart contracts came the possibility of trust-minimized, programmatic transactions of all kinds.

In 2015, the ultimate realization of this possibility came to life with the launch of Ethereum, a cryptocurrency based on the same blockchain data structure as bitcoin but featuring a virtual machine for computing Turing-complete smart contracts. In plain terms, this means that while some cryptocurrencies like bitcoin are only able to support a limited number of ways to program its native currency and contracts, Ethereum is able to natively support any kind of program that can be imagined. This has led programmers and enthusiasts to re-imagine entire industries as trust-minimized series of smart contracts, automating away inefficiencies and disintermediating middle-men along the way.

THE SMART ORGANIZATION

In 2013, entrepreneur and investor David Johnston began describing bitcoin as a "decentralized company" that uses its own internal currency to incentivize rational economic actors to provide value back to the network. The idea was that, using only smart contracts encoded in software, bitcoin was able to coordinate humans and machines to construct a global, trust-minimized, censorship-resistant payments network to rival centralized companies like PayPal or Visa. This view was later expanded on by Dan and Stan Larimer, and then popularized and taxonomized by Vitalik Buterin. The concept of the "decentralized autonomous

organization"--or "DAO" for short--soon sparked the imaginations of many in the growing cryptocurrency movement.

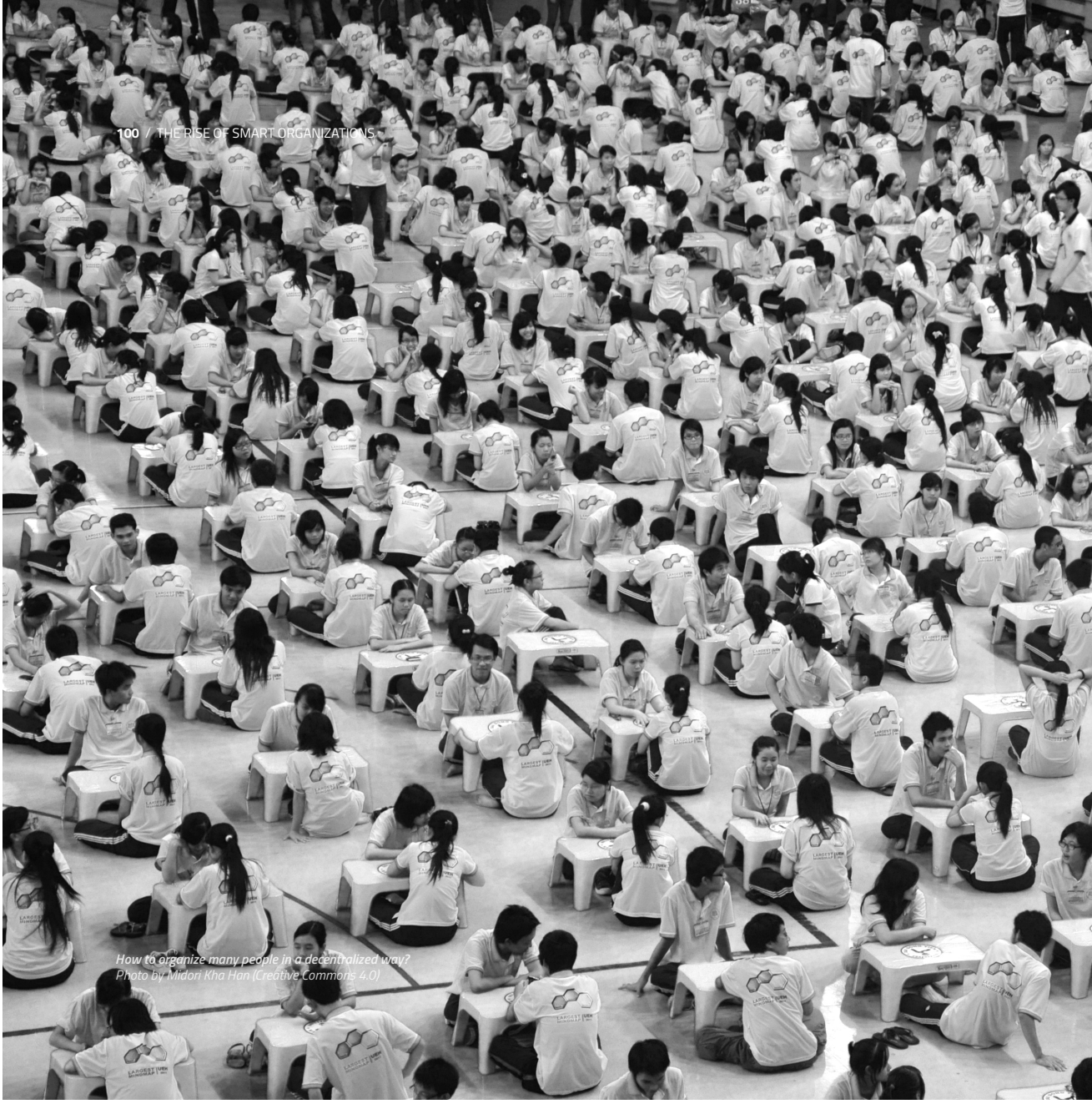
Fast-forward to 2017. Ethereum was not even two years old yet and it had not even been one year since the infamous failure of "The DAO", the first major attempt to launch a DAO on Ethereum. The first alpha version of Aragon, a platform for creating DAOs, was deployed to Ethereum's Kovan test network, and the movement to realize the powerful concept of decentralized organizations was revitalized. With the new, modular architecture of aragonOS, it became possible to program any kind of organization on Ethereum and use Aragon apps to extend their functionality. A new term was needed to describe these organizations, one that encompassed both centralized and decentralized organizations that were built using smart contracts. The "smart organization" was born.

FEATURES OF SMART ORGS

The defining feature of a smart organization is that its operational and governance processes are secured by smart contracts and executed by machines, rather than being secured by paper contracts and executed by humans. With smart contract-based security comes the benefit of increased auditability, reliability, and transparency of transactions in a smart organization.

For example, a traditional organization may have a contract with an employee that says every 30 days, the employee will be paid $5000. Every 30 days, the company will pay the employee--or not, if there are internal power struggles or the financial infrastructure of the company is compromised. If the company doesn't pay as expected, the employee may be forced to pursue a remedy through a lengthy and expensive lawsuit.

A smart organization, by contrast, may use a payroll smart contract to give employees a higher degree of certainty that they will be paid what they are owed on time as specified in the contract. So long as the network that executes the smart contract is online and has capacity to process the transaction, the employee will get paid as expected. An additional benefit is that, because the



How to organize many people in a decentralized way?
Photo by Midori Kha Han (Creative Commons 4.0)

payments are executed autonomously, the employee can choose to get paid at any interval during which transactions are processed by the network; in smart contract networks that use a blockchain, this would mean the possibility of getting paid every block rather than monthly or bi-weekly.

With a higher degree of trust between smart organizations and employees or vendors due to the reliance on smart contracts, transaction costs drop. Less overhead is necessary to ensure the security of transactions, and disputes are resolved efficiently or eliminated completely by the autonomous execution of shared agreements in smart contracts.

Where there is ambiguity, an arbitrator (or decentralized network of arbitrators) can be relied upon to ensure the faithful execution of the smart contract.

These cost-cutting features may lead to smart organizations being more competitive than their traditional counterparts. And all of this can be accomplished without any permission from or interaction with intermediaries such as banks or governments.

Another important benefit of smart organizations is the possibility for distributed governance. Governance over resources or processes that have traditionally been controlled by one or a few people due to technological limitations can be distributed over many entities in a smart organization, up to and including every member of the organization having a say in how resources and processes are governed.

For example, rather than giving only one or two employees authorization to spend funds from the corporate treasury, a smart organization can require authorization from the entire board, or even all shareholders, to make transfers from the main treasury account into smaller, department-owned accounts.

These department-owned accounts can then themselves be controlled by groups of employees in each department, rather than one executive or manager. This can prevent embezzlement and other abuses of power that traditional organizations are vulnerable to.

Continued...

EXAMPLES OF SMART ORGS

Smart organizations may sound great in theory, but what do they look like and how do they behave in practice? While it is still early in their evolution and development, some models are beginning to emerge that show how smart organizations can be governed and used in the real world.

THE DECENTRALIZED PACKAGE MANAGER

In software development, a package manager is a program that is used to deliver the latest version of a piece of software to its users. Traditionally, these package managers are hosted by a centralized organization and operated by a single user account (which is often controlled by a single developer).

This can lead to problems where a compromise of either the package manager host or the package manager operator results in a compromised piece of software being published. Alternatively, package names can be swapped, leading to confusion for users.

A decentralized package manager solves these problems by distributing control over the package manager to members of a smart organization. The members are required to come to consensus according to the smart contracts governing the organization before a new software version can be published to the package manager.

If the package has a name, it can also be owned and governed by the smart organization, again requiring the members to come to consensus before the name can be changed or swapped with a different package. This makes compromise of the software more difficult since multiple entities have to be compromised instead of just one.

THE SMART COMPANY

The joint-stock company was invented as a means of aggregating capital and labor, using profit-sharing contracts to incentivize investment and employment contracts to coordinate workers. In "The Nature of the Firm" (1937), Ronald Coase asserted that firms such as companies arise when the transaction

costs of aggregating labor under one organization are lower than the cost of outsourcing to a third-party supplier. This has led to different types of firms emerging, all of which have been traditionally composed of paper contracts that are executed by humans and secured by a State-run legal system.

As powerful as the modern company is as a form of organization, it still relies on many third parties to operate, including banks to store funds and governments to incorporate the company and resolve disputes. The smart company has employees, investors, suppliers, and customers like a traditional company, but many or all of its resources and processes are secured by smart contracts.

Instead of a bank account, the funds of a smart company are held by a smart contract with programmatic rules over how the funds can be transferred and who is authorized to transfer them. Similarly, the smart company does not use lawyers to draw up a cap table and employment contracts on paper. Instead, smart contracts are used to issue digital shares to shareholders and make payroll so everyone gets paid as expected.

Some processes (such as hiring or dispute resolution inside the company) are still carried out manually, but once a certain outcome involving the transfer of resources inside the organization is required, it can be encoded in a smart contract to ensure reliable and transparent execution.

THE DECENTRALIZED ALTRUISTIC COMMUNITY

Charities today are both loved for their altruistic missions and maligned for the amount of wasteful overhead that some have relative to the benefit they bring to the world. While charities are often required to produce detailed reports about how they spend money, they aren't required to measure the impact they are having, leaving donors wondering whether their money is really being allocated effectively.

There are efforts to increase transparency and measure impact, but donors are still often treated as passive sources of funding rather than active participants in the charity's decision-making.

A Decentralized Altruistic Community (DAC) is a smart organization with a charitable mission such as feeding the homeless, curing diseases, or maintaining a public park. With a DAC, donors get the altruistic benefits of charity with the transparency and control offered by smart contracts.

Donors allocate funds to campaigns proposed by members of the DAC and can see exactly how their funds are being spent. If during the course of the campaign the donor feels that their funds are being misallocated, they can reallocate what is left towards a different campaign using a liquid pledging smart contract. This ensures that there are checks and balances to prevent waste by those carrying out the campaigns.

THE ARAGON NETWORK

What happens if there is a dispute between tokenholders in a smart organization over the legitimacy of a proposal that appears to have popular support? For example, if a malicious entity gains supermajority control of the tokens that govern a smart organization, they could try to pass a proposal that sends all of the funds that belong to the organization to their own private address. Using fiduciary agreements that are judged by a neutral arbitration system, honest minority tokenholders in a smart organization can be protected from a malicious majority trying to pass proposals that are harmful or contrary to the purpose of the organization.

The Aragon Network is a digital jurisdiction for smart organizations. Smart contracts made in this jurisdiction can be arbitrated by jurors in the network's decentralized court system.

The benefit of using the system to arbitrate disputes is that the Aragon Network itself is smart contract-native and can resolve disputes even when all parties are pseudonymous. This is in contrast to regular private arbitration or State-run court systems that require participants to disclose their legal identity and do not yet have a means of interfacing directly with smart contracts.

The Aragon Network is itself a smart organization governed by holders of ANT, the Aragon Network Token. ANT holders set the

rules and parameters that govern the network, vote on finance allocations, and participate as jurors in the court system if they have expertise that's helpful in arbitrating disputes.

Customers of the Aragon Network court who have a lot of value depending on honest outcomes in the court may also hold large amounts of ANT to use as collateral for agreements and to participate in the governance of the court system.

THE FUTURE OF SMART ORGS

It's early days for smart organizations, yet it is already possible for anyone to create them for less cost, in less time, and with greater ease than traditional organizations.

Aragon is a decentralized application that was designed to make the process of creating smart organizations as easy as creating a profile on a social networking app. Programmers can interact with their organizations by interfacing with aragonOS and Aragon app contracts directly, even creating their own apps to enable new functionality in their organizations. And as the Aragon client evolves, it will be possible for users who prefer a graphic interfacesl to have the same level of control over their orgs as a programmer with direct access to the smart contracts would.

For the first time in history, it is possible to experiment with governance and organizational forms at the speed of software rather than the speed of law. Despite the technology being less than two years old at the time of this writing, we are already seeing new forms of organizations with advanced functionality never seen before.

As the tech advances there will be a Cambrian explosion of experimentation in smart organization design, and more people will be freed from the constraints of traditional organizations, unleashing waves of activist and entrepreneurial energy on the world as people begin to exercise their newfound freedom to organize and transact. Smart organizations, with all their power and potential, are here to stay.

Written by John Light

THE PI ZERO W SECURITY CAMERA

One annoying aspect of the 'Internet of Things' is that almost all products rely on shoveling tonnes of data back to random servers owned by the manufacturers. Internet-connected security cameras are a perfect example of this kind of thing, so I thought I'd make my own.

This design uses a Raspberry Pi Zero W and camera module as the brains of the device, and a custom 3D printed case, which is modular, that can be manually panned or tilted to point in the exact direction needed.

MATERIALS

- Pi Zero W
- 5MP Pi Zero Camera Module (Mine is an unbranded, cheap one, Rev 1.3)
- 3D Printed Case Parts
- 2x M2.5 x 8mm Machine Screws
- 4x M2.5 x 12mm Machine Screws
- Suction cup (Optional)

PI ZERO W SETUP

1. First, download the latest Raspbian Lite OS from `raspberrypi.org/downloads` and burn it to a micro SD card.

2. Set up your WiFi and SSH access. Check out the Pi-Hole article in this zine for steps to accomplish this if you don't already know how to set these up.

3. Plug the Zero in, and find its IP address. You can do this either by connecting a monitor and typing `ifconfig`, or check your router admin.

4. Now, SSH into your Zero (replacing the IP address with your own). The default password is `raspberry`:

```
ssh pi@192.168.0.10
```

5. Open up the config utility:

```
sudo raspi-config
```

Now change the default password, and also enable the camera interface.

6. Alright, before we install the software needed, let's do the update command to make sure we have the latest versions of everything:

```
sudo apt-get update
```

7. Now we can install Motion, the utility which we'll use to control the camera. This may take a little while to get through:

```
sudo apt-get install motion
```

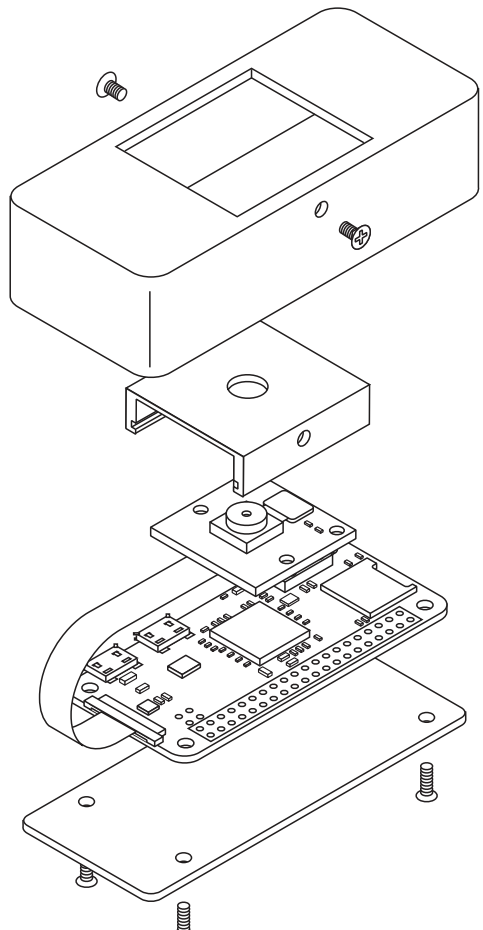
8. Next, we need to turn the Motion daemon on. In the terminal lets open the daemon config file:

```
sudo nano /etc/default/motion
```

Then change `start_motion_daemon=no` to `start_motion_daemon=yes`

Once finished, save and close.

9. Now we have to change a few things in Motion itself to make sure the video streaming works well on the Zero. I've created a custom configuration file to replace the existing one located at `/etc/motion/motion.conf`



This will make a bunch of changes so Motion works well on the Zero W. You can find my `motion.conf` file in the "Pi Cam" folder of the zine dat. The quickest way to get the file onto your Pi is to use a graphical SFTP client and drag it to the target directory.

The config file sets the video to stream at 30fps, with a resolution of 640x480. Alternatively, you can check out the config file for yourself and change it to whatever best suits your situation.

The final thing to do is edit the file `/etc/init.d/motion` so the correct drivers are loaded during the boot process:

```
sudo nano /etc/init.d/motion
```

Then add the following line after the first line that says "chown" a bit further down:

```
sudo modprobe bcm2835-v4l2
```

Save and exit, reboot the Pi, and that's the software side of things complete. Motion has tonnes of options, so look through their docs if you want to set your cam up differently.

HARDWARE SETUP

1. Attach the camera FPC cable to the Pi Zero.

2. Slide the camera, face down, into the camera holder printed piece.

3. Now place the camera holder into the case, with the FPC end directed towards the bottom.

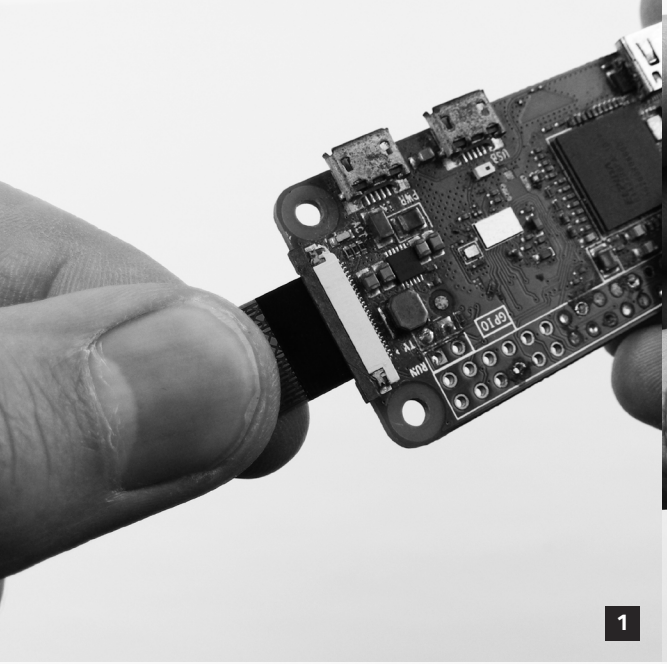
4. Screw the camera holder in, with the 2x M2.5 x 8mm screws.

5. Place the Pi Zero W in the case, with the Micro SD Card side pointing towards the top.

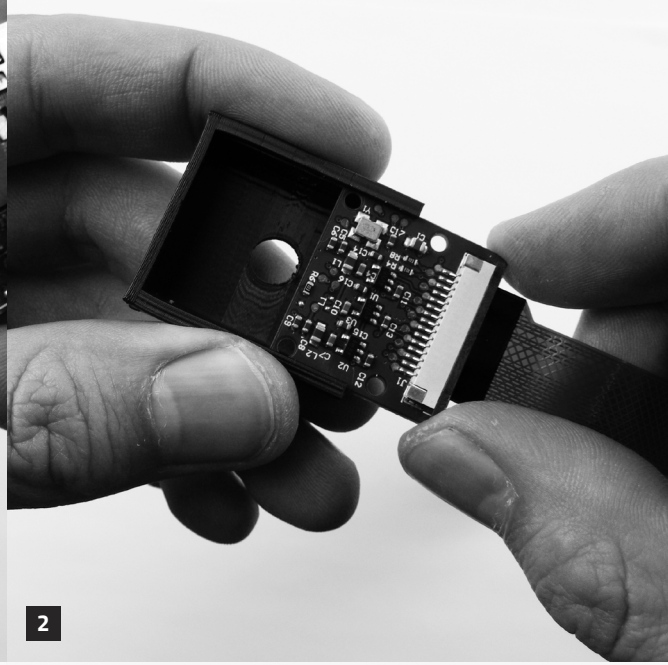
6. Secure the backplate with the 4 screws.

7. Now you can choose the bracket or hook attachment to fix it into place. I added a suction cup to stick mine to a window.

8. Once affixed, simply power it on, and it should automatically begin streaming. To view the live video, use a browser to connect to `192.168.0.10:8081` (replacing the IP with yours) from any phone, tablet or computer connected to your network.

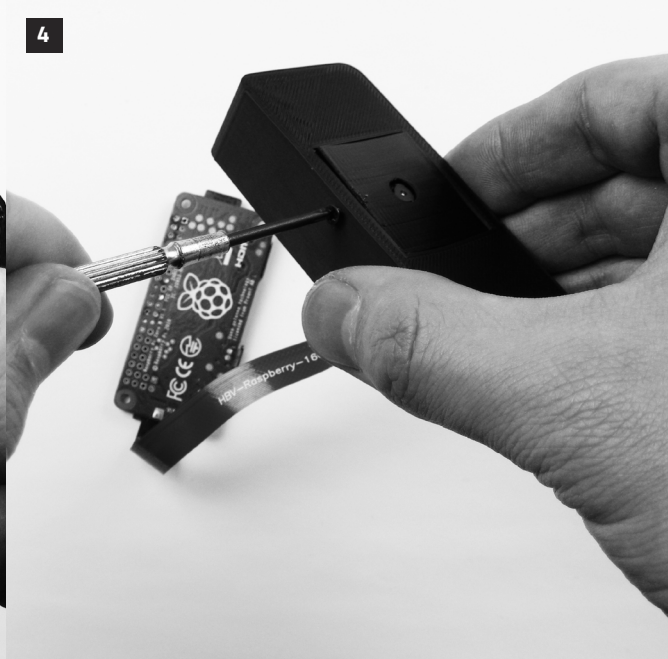


1



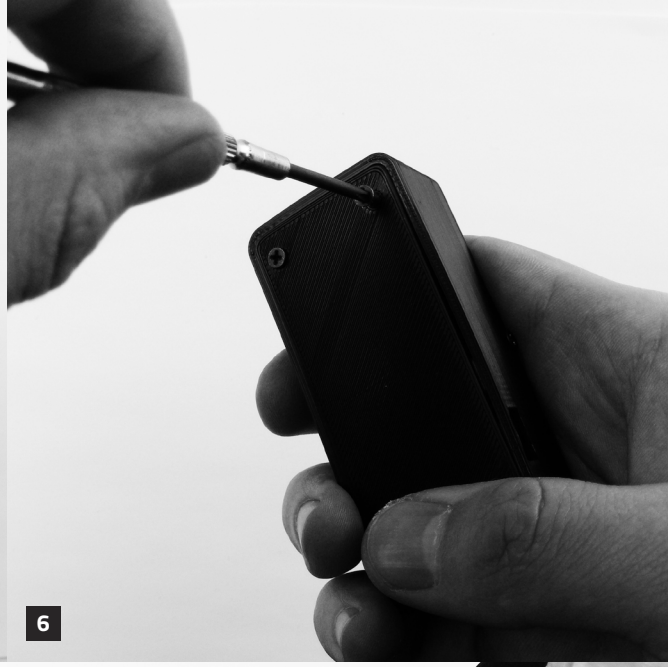
2

3



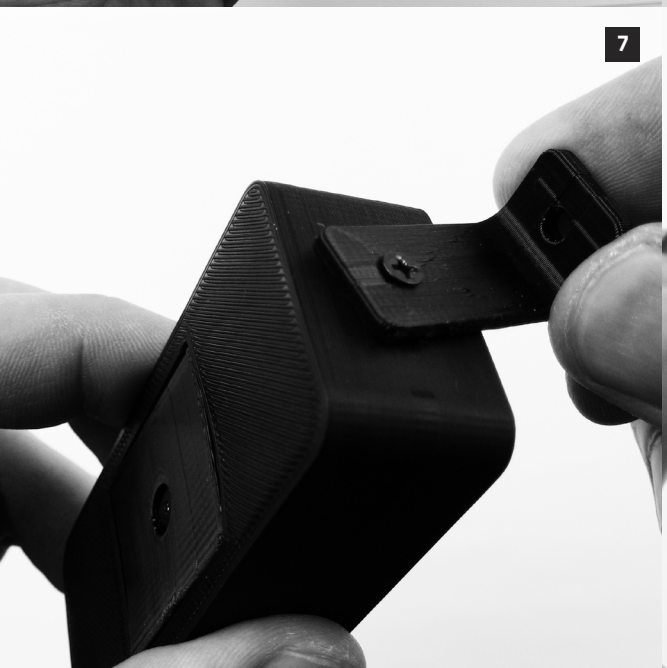
4

5

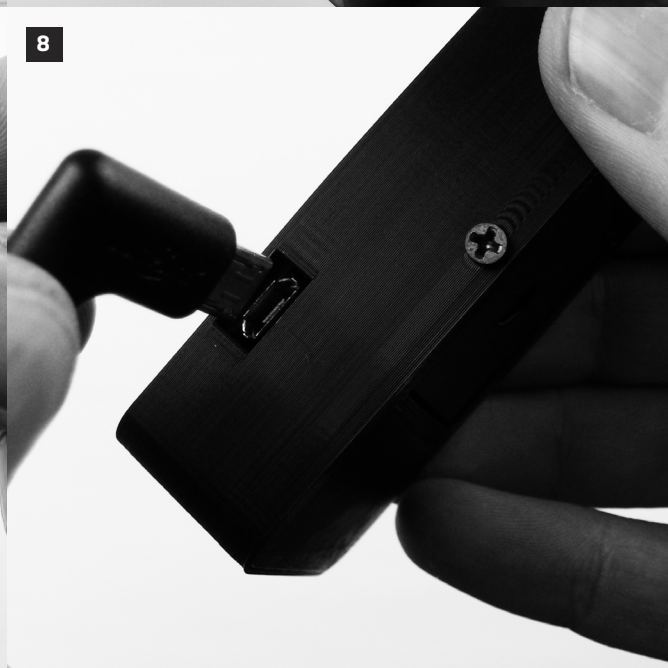


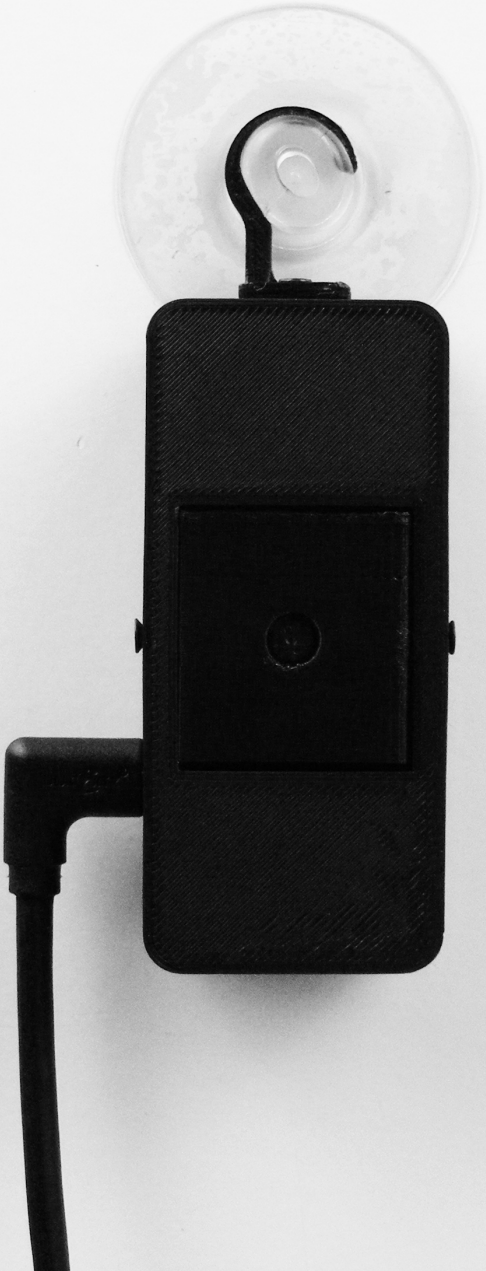
6

7



8





OPEN BAZAAR: DECENTRALIZING ONLINE COMMERCE

Back in 2014 a small group of people, myself included, didn't like seeing online commerce so tightly controlled and monitored. We decided that we needed a platform for trade that isn't controlled by companies or any middlemen at all. That's why we built OpenBazaar.

OpenBazaar is a decentralized marketplace. It's a network of people running peer-to-peer software--similar to BitTorrent or Bitcoin--who connect directly to each other in order to buy and sell goods and services. Since everyone in the network is connected peer-to-peer, there are no middlemen involved, and that means there is no one to collect fees, collect user data, or censor trade.

We decided to build OpenBazaar to give the user as much control over their own data as possible. OB only stores data locally on your computer, so users don't create accounts, and aren't forced to share their identities. All communications between users is encrypted for added privacy too.

We knew from the start that the online commerce platforms weren't the only companies which monitored and censored user transactions.

Payment processors such as PayPal or credit card companies do the same thing. That's why payments in OpenBazaar use cryptocurrencies including Bitcoin, Bitcoin Cash, Litecoin, Zcash and Ethereum.

Dispute resolution and escrow are also built into the platform. Buyers and sellers agree on moderators (escrow agents) who will settle disputes that would arise if something goes wrong. These moderators are available on an open marketplace, and are able to charge fees in order to settle disputes.

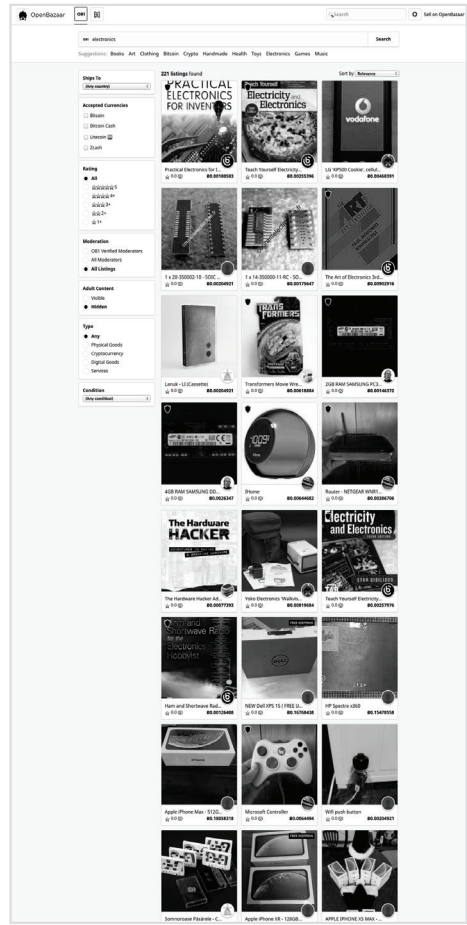
The software is open source, and has been through several major revisions over the past five years. The latest versionis built on top of IPFS so all data is distributed across the network, and is highly censorship-resistant.

While we built this for desktop initially, there's now a mobile application you can use to access the OpenBazaar network called "Haven," which is available on iOS and Android.

It's been an exciting journey over the past few years trying to build and gain adoption for the world's largest decentralized marketplace. Since anyone can sell anything they want, there is an extremely diverse range of products and services for sale. People have even used OpenBazaar to rent out their apartments and offer taxi services instead of using Airbnb or Uber. They've sold their homemade hot sauces and their DIY electronics creations.

Several hundred thousand people have downloaded and run OpenBazaar over the past few years. You can view what is on the network directly in a browser by visiting `openbazaar.com`, or you can download the software and become part of the network yourself by visiting `openbazaar.org`.

Written by Sam Patterson



MNT REFORM: AN OPEN SOURCE DIY ARM LAPTOP

Ever since the Raspberry Pi launched in 2012, people have been trying to bodge and couple SoCs with various lapdocks to create something akin to a makeshift laptop. While many of these designs worked and resulted in usable systems, they didn't necessarily work *well*, or were otherwise fragile and roughly put together. A lot of the ideas were there, but nobody was actively trying to build a complete laptop from the ground up using an ARM SoC. That would be crazy, right?

Lukas F. Hartmann is changing all of that with Reform, an open-source laptop he's developing under his MNT Research company based in Berlin, Germany. "Before starting Reform, I released an open-source FPGA based graphics card for vintage Amiga computers (VA2000)," Lukas explains. "This was my first real product with open-source components and at the same time my first hardware release. Making hardware feels a bit like reconnecting with my childhood where I experimented with soldering and read electronics books without understanding anything, but I always loved the aesthetics."

MNT Research, lead by Lukas embarked on the Reform project in 2017. By December of 2018, ten beta units were shipped out to curious, adventurous people who had pre-ordered the experimental machines and ultimately got the chance to test out Lukas' dream-come-true. Currently, Reform development is ongoing to make design improvements and take in feedback from the beta design.

"I wanted a device that is easy to understand, comes with schematics (like a Commodore 64 did), invites you to tinker and customize, and would be a great typing machine," Lukas says.

HARDWARE

The real star of Reform is, in my opinion, the custom chassis that houses all of the components. Borrowing design queues from early home microcomputers (like the Sinclair


NAILFUN
ISOPROPANOL
99,9%

ZX Spectrum, Commodore 64 and Amiga 500) as well as early laptops (I feel a strong influence from early IBM ThinkPads, myself), the chassis aims to be the core of a modular system, allowing for parts to be easily repaired, replaced or upgraded as needed for years to come.

"Case parts should be replaceable using a 3D printer and electronics fixable with a soldering station," Lukas explains. Reform invites users to open up the hardware to hack and experiment. Mess up something while making a modification? Print out a replacement. At home. For free.

The brain of the system is the NXP i.MX6 QuadPlus SoC, boasting four ARM Cortex-A9 cores running at up to 1.2 GHz. The i.MX6 also offers a Vivante GC2000 GPU that has completely open-source drivers within mainline Linux (etnaviv) and OpenGL (mesa). Lukas shares that he "wanted to use a CPU/SoC that is as open as possible and doesn't require any closed source drivers."

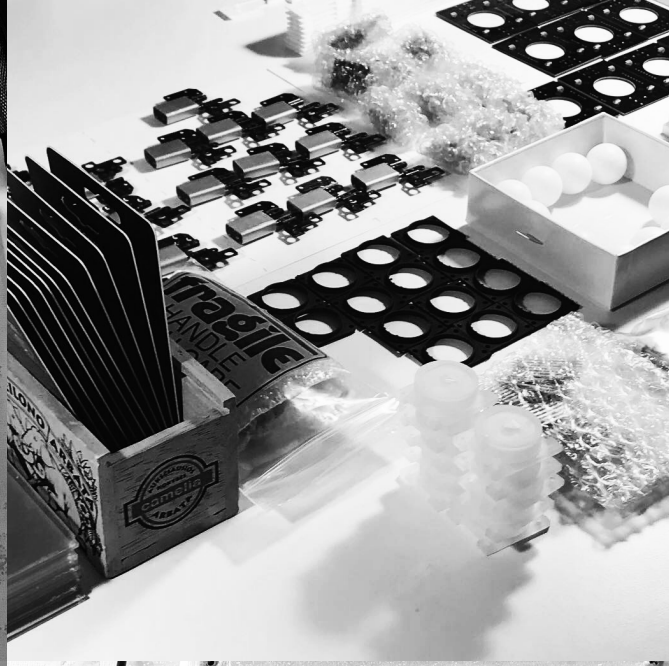
Each part of the machine is intended to be as open as the chassis, which the i.MX6 further illustrates through its lack of needing various NDAs or registration requirements.

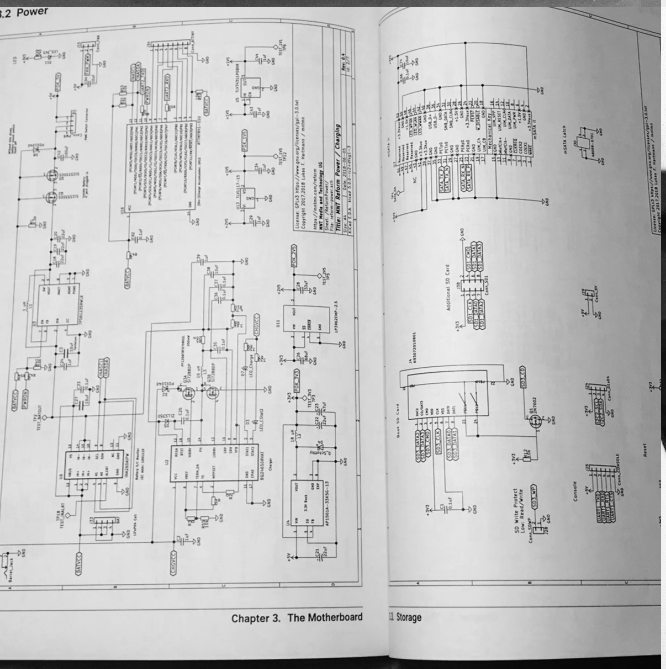
5x USB2.0, HDMI, 1000Mbps Ethernet, 3.5mm audio, mPCIe, WWAN, mSata, SPI, I2C, and GPIO interfaces are all offered from a custom motherboard that also houses 4GB of DDR3 RAM, a SIM card slot, and a microSD card slot (bootable). The battery, currently composed of a single 10Ah LiFePo4 cell, is a safe and durable power-source that can be charged with a 5-volt DC adapter.

The display is a 11" IPS panel that can output a 1366x768 resolution. The swappable keyboard is built using Cherry ML switches, while the pointing device is a custom trackball with two buttons (also using Cherry ML switches), a rarity in laptops these days.

SOFTWARE

The i.MX6 is completely blob-free, and fully supported by mainline Linux. Currently, a Debian-based image is available for Reform, but any other Linux distribution should be easily supported. Other OS ports are planned.




3.2 Power
Chapter 3. The Motherboard
3.3. Storage


Net: Board Net Initialization Failed
No ethernet found.
starting USB...
USB0: USB EHCI 1.00
scanning bus 0 for devices... 1 USB Device(s) found
USB1: USB EHCI 1.00
scanning bus 1 for devices... 3 USB Device(s) found
scanning usb for storage devices... 0 Storage Device(s) found
Hit any key to stop autoboot: 0
Welcome to Reform.
=>

PROJECT GOALS

Reform is built to be the ultimate hackable, repairable, and customizable laptop computer. Though the processor may not be comparable to a top-of-the-line amd64, a lot of care was taken to balance performance with privacy and openness. "I don't want to pretend that Reform is better than all other laptops," Lukas elaborates.

"What I wanted is a strong alternative, a machine that goes against many established trends. Miniaturization makes many modern laptops hard to service for average people, or you get soldered-in batteries or SSDs. It is hard to understand what the individual components do."

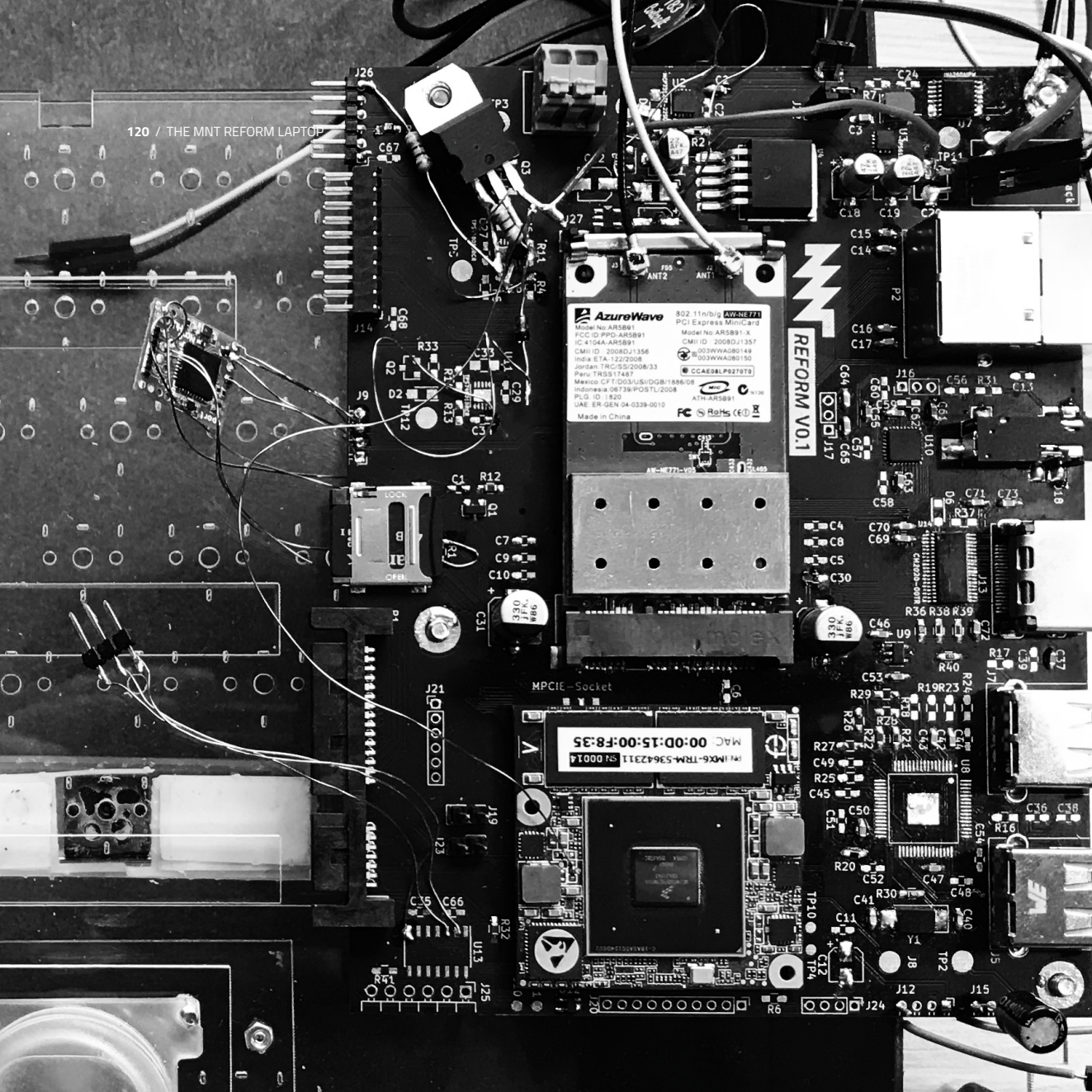
Valuing function over form is important for Lukas. "MacBooks are treated like expensive fashion accessories that you wouldn't crack open and put your own modifications in," he says, adding that "the ergonomics of keyboards are suffering from the dogma of making everything ultra slim." That said, though Reform focuses on openness, it exhibits good design as well.

In the spirit of security and privacy, Reform ships with no built-in camera or microphone. You don't have to worry about these components as attack vectors, unless you decided to add them yourself later.

FUTURE

Many design improvements are planned for the chassis in the next Reform iteration, including increased ergonomics, a new keyboard layout, cooling changes, and consolidated mechanical switches. Additionally, there are plans for a multi-cell battery and support for more screen options (as people may want to use a higher resolution display, or even E-Ink).

Considerations have been made to change the SoC to something more performant, such as the more recent i.MX8, but it currently utilizes binary-blobs, has poor documentation, and incomplete GPU drivers. Alternative options, like a RISC-V chip, are not available at a low enough price-point to make them economically feasible. All hope for RISC-V is not lost, however. "As soon as affordable and


AzureWave
AW-NE771
REFORM V0.1
MPCIE-Socket

fast chips will become available, I'd like to move to the RISC-V platform (which has an open instruction set architecture) and do some experiments of connecting it to our open source graphics tech developed for VA2000 and now ZZ9000," Lukas shares.

Having ports of other operating systems besides Debian Linux is also in the pipeline according to Lukas. "Next to the BSDs, I'd also like to see ports of Plan9, RiscOS, and BeOS. I chose Debian GNU/Linux as the first OS because this is what I'm most familiar with. I also wrote my own toy OS 'Interim' that I'd like to port one day."

We may also see Reform hardware outside of the laptop form factor as Lukas discloses that "Reform is supposed to be a family of open devices of all sizes."

CONCLUSION

The Reform laptop project is a fresh take on creating an open computer, and it has already produced a tangible, working beta unit. While the current specifications should be suited to developing, writing, and general computing tasks, we may one day see the Reform platform expanded for other purposes.

There is no current release date for the next iteration of the Reform laptop, but the price-point is likely to be similar to that of the beta unit (and the initial estimate) at around €500-700.

MNT Research has also recently moved into a new location. Lukas explains, "the first version of Reform was mostly assembled in our (Greta & my) shared flat, but we just moved to a real studio space in Berlin Friedenau to focus on OSHW and other projects (like clothing)." With any luck, a larger space will allow more room for Reform to grow.

Reform is still a work in progress, check for updates at `https://mntmn.com/reform`

Written by Mike Dank

(Photos provided by Lukas Hartmann)

DECIPHERING THE LEXICON

With constant development of decentralized networks and digital currencies, it is easy to get lost when trying to figure out exactly how something works or what it is trying to accomplish.

There are a lot of concepts, new and old, that may be considered common knowledge when they are actually anything but.

Here are just a few terms or phrases that are easy to understand, but often glossed over in documentation.

Written by Mike Dank

Asymmetric Cryptography. Also known as public-key cryptography, Asymmetric Cryptography schemes have encryption and decryption performed by separate keys.

Data may be encrypted with a user's public key (generally available to everyone), so it can only be decrypted by that user holding a corresponding private key (only known by the recipient). Conversely, data may be encrypted with a user's private key (and decrypted by anyone with the user's public key) to digitally sign data and show it could only have come from that user.

Blockchain. A growing linkage of blocks, where each block contains a cryptographic hash of the previous block, a timestamp, and any transactional data.

Blocks are cryptographically linked using Cryptographic Hash Functions, and generally rely on a Proof of Work algorithm.

Bloom Filter. A space-efficient data structure, usually used for quickly looking up data. Bloom Filters are commonly used to combat the space and time requirements to traverse a large set of the data.

Ciphertext. Encoded data that is not user-readable. Ciphertext may be intercepted by bad actors in a network, but it cannot be deciphered by anyone but the intended recipient.

Cryptographic Hash Function. A mathematical algorithm that converts one piece of data into another of fixed size.

Cryptographic Hash Functions are designed to be "one-way functions" meaning it is mathematically infeasible to invert the resulting hash back into any original data. This makes the functions suitable for proving work accomplished between multiple actors, without revealing a solution upfront.

DApp. A decentralized application. DApps are run by many users on a Decentralized Network, and store data on a Blockchain. They are designed to be trustless, autonomously managed, and devoid of a single point of failure.

DAO. A decentralized autonomous organization. DAOs are organizations built to follow a specific set of rules, carried out via software.

Usually financially oriented, rules are programmed to broker transactions, which are recorded in a Blockchain.

DAOs are meant to be independent and transparent; their rules are public knowledge, and they respond to shareholders instead of a central government.

Decentralized Network. A network topology where multiple servers are linked together, allowing clients to connect to any server and still be within the same network.

Distributed Hash Table. A paired key/value data structure, usually used in routing algorithms for Distributed Networks. Nodes in a network might keep a table of their local peers, but have no other information about the rest of the network.

If a node needs to send data to another in a network that it isn't directly peered with, it can consult its local table to find a peer that is closet (based on address) and forward the message.

That next peer then repeats the same process, which continues until the data reaches the destination. No single node has a complete view of the whole network.

Distributed Network. A network topology where all actors connect and communicate with one another, forming a peer-to-peer network. Each actor is both a client & a server.

End-to-End Encryption. A system where data is encrypted by a sender and decrypted only by the recipient(s). The data will remain encrypted and tamper-proof no matter what network, server, or other third-party it might pass through as only a recipient will possesses the decryption key.

Federated Network. A network topology consisting of smaller, more centralized self-governed organizations (usually one server and many clients) that elect to share data with one another.

Key Exchange. A method of transferring encryption keys between two or more parties. For two actors to communicate in an encrypted conversation, they must each share keys to properly encrypt and decrypt messages to one another. Some key exchange methods, like Diffie-Hellman, allow for keys to be securely exchanged over an insecure channel.

Merkle Tree. A tree-like data structure, containing chunks of data chained together. Every node in the tree is either a "leaf," labeled with the hash of a data block, or a "non-leaf," labeled with the hash of the labels of its child nodes. Merkle Trees are used to verify large amounts of data.

Mesh Network. A network topology where all infrastructure nodes connect directly (without hierarchy) to as many other nodes as possible. Each node works with others to route data through the network.

Onion Routing. A layered encryption routing method where data in a network is encrypted multiple times by the originator, using public keys of nodes the data will pass through on its way to a recipient. When the data is sent out, it is partially decrypted by the next node in the chain, revealing only enough information to pass the data on until it reaches its recipient, who can decrypt it as intended.

Oracle. Middleware that connects real world events to a Blockchain through Smart Contracts. If a certain set of criteria in the real world is met, a Smart Contract may execute a transaction within a Blockchain.

Overlay Network. A network that is securely encapsulated within a larger network. An overlay network may be private in nature, and tunnel over a more public network like the Internet.

Proof of Stake. An algorithm found in cryptocurrencies where the creator of a block is chosen by a stake in the network, sometimes coupled with random selection.

Proof of Work. An algorithm found in cryptocurrencies where a block is mined by a computationally intense process used to validate transactions

Smart Contract. A self-executing software contract, that executes when the terms of an agreement between parties are satisfied.

PRODUCT DIRECTORY

COMPUTERS & SERVERS

CASA NODE

Plug and play, BTC and Lightning node. RPi 3b+ design with 1TB HDD.

$300.00
https://keys.casa

NODE MINI SERVER

Turn your Raspberry Pi 3b+ into a P2P node with this open source adapter.

£TBD
https://n-o-d-e.shop

TECHNOETHICAL X200S

Refurbished Thnkpad X200s laptop with custom Libreboot BIOS.

€478.00
https://tehnoetic.com

VIKINGS D8

FSF certified Linux workstation, with multiple configuration options.

FROM €775.00
https://store.vikings.net

BITSEED 3

Plug and play Bitcoin full node. Fanless design with 4GB RAM and 1TB HDD.

$359.00
http://bitseed.org

NODE NANO SERVER

Pi Zero W wall outlet adapter. Turn your Zero into an always-on mini server.

£19.00
https://n-o-d-e.shop

TECHNOETHICAL T400

Refurbished Thnkpad T400 Laptop with Libreboot BIOS and GNU/Linux-Libre OS

€568.00
https://tehnoetic.com

TECHNOETHICAL D16

Linux workstation with 2x AMD 2.6Ghz CPUs, 128GB RAM, Libreboot BIOS

€3700.00
https://tehnoetic.com

VIKINGS X200

Refurbished Thinkpad X200 laptop with custom Libreboot BIOS. FSF Certified.

FROM €245.00
https://store.vikings.net

LIBREM 13 V4

Modern Linux laptop. Intel i7 CPU, up to 16GB RAM, hardware switches + more.

FROM $1399.00
https://puri.sm

GNUBEE PERSONAL CLOUD 2

Open NAS server. 512MB RAM. Space to attach up to 6 3.5' SATA drives.

$249.00
http://gnubee.org

SKYMINER

Node for Skycoin alt Internet. 8 CPUs, 16GB RAM, & OpenWRT switch built in.

$1999.00
https://store.skycoin.net

DIGITAL SECURITY

USB CHARGING KEYCHAIN

Open hardware keychain that stops USB data transfer, for safe charging.

£8.75 (2 pack)
https://n-o-d-e.shop

HELIOS 4

Open NAS server. 2GB RAM included, and space for 4 external SATA drives.

$195.24
https://kobol.io

MNT REFORM

Open source and modular DIY laptop. ARM CPU, 4GB RAM, 3D Printed Parts

$TBD
http://mntmn.com/reform

TECHNOETHICAL X200S

Refurbished Thnkpad X200s laptop with custom Libreboot BIOS.

€478.00
https://tehnoetic.com

C@RD MARK II

Low tech card for generating and recalling complex passwords.

$6.99
http://russtopialabs.bigcartel.com

NFC KEY

2x 8KB chips for storing data. Only transmits when button is pressed.

£15.00
https://n-o-d-e.shop

WEBCAM SLIDER STICKER

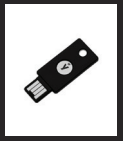
Cover the cameras on your devices. Can slide open whenever needed.

£3.50 (3 pack)
https://n-o-d-e.shop

RFID WALLETS

Protect your tap to pay cards from passive activation with these wallets.

£2.50 (5 pack)
https://n-o-d-e.shop



YUBIKEY 5 NFC

USB/NFC authenticator. Supports multiple authentication protocols.

$45.00
https://www.yubico.com

PURISM KEY

USB authenticator designed at verifying BIOS info, and decrypting disks.

$59.00
https://shop.puri.sm

MOOLTIPASS MINI

Smartcard based password manager / auth. Works with multiple platforms.

$79.00
https://themooltipass.com

P@SS MARK II

Low tech keychain for generating and recalling complex passwords.

$24.99
http://russtopialabs.bigcartel.com

COMMUNICATIONS

GL-B1300 ROUTER

Dual-band homw mesh router. Comes with OpenWRT firmware.

$89.00
https://store.gl-inet.com

GL-MIFI 4G ROUTER

4G Router that transfers 3G/4G LTE signal to wifi at 150mbps. OpenWRT

$109.00
https://store.gl-inet.com

WNDR3800 ROUTER

Re-branded Netgear WNDR3800 with LibreCMC firmware.

€35.00
https://store.vikings.net

LIMESDR

Open source SDR. Send/receive UMTS, LTE, GSM, LoRa, Bluetooth, Zigbee, RFID

`$299.00`
`https://limemicro.com`

HACKRF SDR

Open source software defined radio. Send/receive from 1MHz to 6GHz.

`$299.00`
`greatscottgadgets.com/hackrf`

uBITX

Arduino-based high frequency radio transceiver kit. 3-30MHz Operation.

`$129.00`
`http://ubitx.net`

MOBILINKD TNC

Battery powered packet radio TNC. Connects to Android devices over BT

`$64.95`
`https://store.mobilinkd.com`

CRYPTOCURRENCY HARDWARE

OPENDIME

USB credsticks that let you transfer Bitcoin offline and in person.

`$39.94 (3 Pack)`
`https://opendime.com`

COLDCARD WALLET

BIP174 Bitcoin wallet that can be used completely offline for it's entire lifecycle.

`$69.94`
`https://coldcardwallet.com`

GENESIS1

2-way BTC ATM. Buy / Sell BTC for cash. Includes high capacity cash dispenser.

`$14500.00`
`https://bitcoinatm.com`

LAMASSU SINTRA

2-way Cryptomat for buying and selling currencies (BTC, BCH, ETH, DASH, ZEC)

`€7500.00`
`https://lamassu.is`

LAMASSU DOURO II

The original Bitcoin ATM. Buy / Sell BTC, BCH, ETH, DASH, ZEC

`€5200.00`
`https://lamassu.is`

COVAULT MODEL X

Freestanding Bitcoin ATM, with 1500 bill capacity.

$4999.00
https://covaultbtm.com

CRYPTOSTEEL

Steel backup for your crypto seed. Resistant to fire, water, and corrosion.

$79.00
https://cryptosteel.com

KEEPKEY

Multicurrency hardware wallet, with built in Shapeshift integration

$129.00
https://www.keepkey.com

LEDGER NANO X

Bluetooth enabled hardware wallet. Can store up to 100 assets simultaneously.

£109.00
https://ledger.com

GENETIC ENGINEERING KIT

The Odin's genetic engineering home lab kit. Includes DIY CRISPR kit.

$1699.00
http://the-odin.com

TREZOR ONE

Hardware wallet that supports more than 700 different cryptocurrencies.

€83.49
https://shop.trezor.io

TREZOR MODEL T

All in one wallet that stores multiple currencies, passwords and digital keys.

€180.29
https://shop.trezor.io

CRYPTO KEY STACK

Low tech steel wallet. Use the included engraver to record your 12 word seed.

$45.00
https://cryptokeystack.com

AUGMENTATIONS

CYBORG BEAST KIT

Parts kit for the 'Cyborg Beast' 3D printed prosthetic hand.

$35.00
http://enablingthefuture.org/shop

ATOUN MODEL Y

Powered exoskeleton. Alleviates strain when the user lifts heavy objects.

$POA
http://atoun.co.jp

EMOTIV INSIGHT

5 Channel brain computer interface, designed for self-quant & device control.

$299.00
https://www.emotiv.com

VIRTUAL REALITY

HAPTX GLOVES

Feel size, weight, and impact in VR. Applies up to 4 lbs of force feedback.

$POA
https://haptx.com

CYBERITH VIRTUALIZER

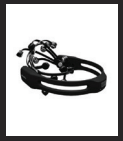
Omnidirectional treadmill for VR. Steam "touchpad locomotion" support.

$POA
https://www.cyberith.com

PAIN MANAGEMENT KIT

A collection of materials biohackers use to manage pain during procedures.

$39.00
https://dangerousthings.com



EMOTIV EPOC+

14 Channel mobile EEG designed for human brain research & device control .

$799.00
https://www.emotiv.com

DEXMO

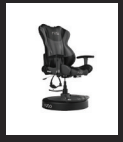
Wireless haptic glove system that lets you feel VR/AR worlds.

$3000.00
https://dextarobotics.com

VRGLUV

Accurately sense users' hands and finger movement in real time.

$POA
https://www.vrgluv.com



ROTO VR CHAIR

Motorized chair with 360 degree movement, and head tracking.

$999.00
https://www.rotovr.com

UNLIMITEDHAND

Sensor which attaches to the forearm, and reads hand/arm muscle signals.

$319.99
http://unlimitedhand.com/en

TESLASUIT

Full body VR clothing. Features haptic feedback, motion capture, temp control.

$POA
https://teslasuit.io

3DRAG KIT

Open source dual extruder 3D printer kit. 170x200x200mm build size.

€516.50
https://store.open-electronics.org

SQUINK PCB PRINTER

Single side PCB printer. Print traces, dispense solder, and pick/place.

$3999.00
https://botfactory.co

PRUSA I3 MK3S KIT

Excellent printer. 250x210x210mm build size. Supports almost all filament

$906.29
https://shop.prusa3d.com

VRFREE GLOVE SYSTEM

Mobile VR gloves with hand/finger tracking & full degrees of freedom.

$390.00
http://www.sensoryx.com

PERSONAL MANUFACTURING

LULZBOT MINI 2

One of the original open source 3D printers. 160x160180mm build size.

$1500.00
https://lulzbot.com

CIRQOID

Mill PCBs, dispense solder paste, and populate boards with this.

$2000.00
https://cirqoid.com

P400 CUTTING PLOTTER

Hot wire cutter used to accurately cut polystyrene sheets/blocks.

€249.00
https://store.open-electronics.org

NOMAD 883 PRO

Desktop CNC machine. 8x3" Work area. Cut/engrave plastics, woods & metals.

$2499.00
http://shop.carbide3d.com

MINIMILL KIT

Open source CNC. 5x5" Work area. Cut/engrave plastics, woods & metals.

$353.75
https://openbuildspartstore.com

LEAD CNC 1010 KIT

Accuarate lead screw based CNC mill with large 40x40" work area.

$946.30
https://openbuildspartstore.com

C-BEAM KIT

Open source CNC mill with 350x280mm work area.

$521.79
https://openbuildspartstore.com

CREALITY ENDER 3 KIT

Open source 3D printer kit with 220x220x250mm build size.

$209.00
https://creality3donline.com

CREALITY CR-10 KIT

Aluminum based 3D printer kit with large 300x300x400mm build size.

$399.00
https://creality3donline.com

X-CARVE

Modular, customizable CNC mill, with highly scalable work area.

$1349.00
https://inventables.com

BOOKS

INFORMATION SECURITY FOR JOURNALISTS

Practical information security handbook aimed at journalists, but useful for all.

https://arjenkamphuis.eu/book

WIRELESS NETWORKING IN THE DEVELOPING WORLD

Free ebook about designing,and building low-cost wireless networks

http://wndw.net

MAKING IT

Manufacturing techniques for product designers/inventors. Valuable knowledge for hardware hackers.

`http://chrislefteri.com`

CRYPTOGRAPHY: A VERY SHORT INTRODUCTION

A great overview of the various fundamental concepts in cryptography.

`http://amazon.com`

THE ART OF ELECTRONICS (3RD EDITION)

The ultimate handbook for electronics designers. Covers almost everything.

`https://artofelectronics.net`

HACKING THE XBOX

Bunnie Huang's masterful book on reverse engineering, complete with detailed, step-by-step examples.

`https://nostarch.com/xboxfree`

HACKING: THE ART OF EXPLOITATION (2ND EDITION)

A no-nonsense guide to understanding various hacking techniques.

`https://nostarch.com/hacking2.htm`

THE KNOWLEDGE

How to rebuild the world after an apocalypse. Described as the blueprint for rebooting civilization.

`http://lewisdartnell.com`

MESSING AROUND WITH PACKET RADIO

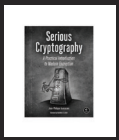
Free zine on how to make TCP/IP ('internet') connections using radios.

`http://archive.org`

COMPUTER LIB DREAM MACHINES

1974 books by Ted Nelson outlining the computer revolution.

`http://amazon.com`



SERIOUS CRYPTOGRAPHY

A technical book for understanding and implementing modern cryptography methods.

`https://nostarch.com/seriouscrypto`

OPEN SOURCE DIRECTORY

P2P / COMMUNICATIONS

Adamant	*Blockchain-based anonymous messenger*	`https://adamant.im`
Aether	*Decentralized pulic communities*	`https://getaether.net`
Afari	*Blockstack based social network*	`https://afari.io`
Akasha	*IPFS/Ethereum based social network*	`https://akasha.world`
Matrix	*An open network for secure P2P communications*	`https://matrix.org`
RetroShare	*Cross-platform, secure, P2P communications*	`http://retroshare.us`
Ricochet	*Encrypted instant messaging routed through TOR*	`https://ricochet.im`
Secure Scuttlebutt	*Decentralized secure gossip platform*	`https://scuttlebutt.nz`
Sigle	*Create decentralized blogs*	`https://sigle.io`
Tox	*P2P audio, video and text communications*	`https://tox.chat`
Unwalled Garden	*Dat-based social network*	`github.com/beakerbrowser/unwalled.garden`
VoluntaryNet	*Browser-based P2P social network*	`https://voluntary.net`

P2P / DATA

Airlock	*IPFS/Ethereum based file storage/sharing*	`https://github.com/slothbag/Airlock`
ClearSkies	*Dropbox-like file sync program*	`https://github.com/jewel/clearskies`
Dat Project	*File storage/syncing protocol*	`https://datproject.org`
IPFS	*P2P data distribution and storage*	`https://ipfs.io`
OnionShare	*TOR based tool for anonymous file sharing*	`https://onionshare.org`
P2PVPS	*Decentralized VPS network*	`https://p2pvps.org`

P2P / DATA

Radicle	*P2P stack for code collaboration*	`http://radicle.xyz`
Storj	*Decentralized cloud storage*	`https://storj.io`

P2P / ENTERTAINMENT

Bittube	*IPFS based video platform*	`https://bit.tube`
Decentraland	*Decentralized virtual reality world*	`https://decentraland.org`
D.Tube	*Steemit/IPFS base video platform*	`https://d.tube`
LBRY	*Community run digital marketplace*	`https://lbry.io`
Livepeer	*Ethereum based video platform*	`https://livepeer.org`

P2P / MONEY

Bisq	*Decentralized cryptocurrency exchange*	`https://bisq.network`
Bitcoin	*A peer-to-peer electronic cash system*	`https://bitcoin.org`
Lightning Network	*Scaling network for Bitcoin transactions*	`https://lightning.network`
Litecoin	*Peer-to-peer internet currency*	`https://litecoin.org`
Monero	*Privacy-focused P2P digital currency*	`https://getmonero.org`
Open Bazaar	*A decentralized ecommerce platform*	`https://openbazaar.org`
Origin	*Marketplaces on the blockchain*	`https://originprotocol.com`

P2P / NETWORKING

Althea Mesh	*Incentivized mesh networking*	`https://althea.org`
Cjdns	*E2E encrypted IPv6 mesh networking*	`https://github.com/cjdelisle/cjdns`
DN42	*Decentralized private networking*	`https://dn42.net`
Hyperboria	*P2P, encrypted private networking*	`https://hyperboria.net`
LibreRouter	*Project to design multi-radio routers*	`https://librerouter.org`
Open Garden	*Decentralized wifi sharing platform*	`https://opengarden.com`
RightMesh	*Ad-hoc mobile mesh networking platform*	`https://rightmesh.io`
Yggdrasil	*E2E encrypted IPv6 mesh networking*	`https://yggdrasil-network.github.io`

P2P / NEW INTERNET

Beaker Browser	*Browser for Dat based decentralized web*	`https://beakerbrowser.com`
Blockstack	*Platform for building decentralized apps*	`https://blockstack.org`
Enigma	*Privacy layer for the decentralized web*	`https://enigma.co`
Ethereum	*P2P network for smart contracts*	`https://ethereum.org`
Maidsafe	*Creating a new alternative internet*	`https://maidsafe.net`
Namecoin	*Blockchain based naming system*	`https://namecoin.org`
Solid Project	*Project to create a new decentralized web*	`https://solid.mit.edu`
SkyWire	*Decentralized platform to build a new internet*	`https://skycoin.net/skywire`
Swarm	*Serverless hosting, and P2P data distribution*	`https://swarm.ethereum.org`
YaCy	*Decentralized, user-run search engine*	`http://yacy.net/en`

P2P / NEW INTERNET

Zeronet	*Open, free and uncensorable websites*	https://zeronet.io

AUGMENTATIONS

Fable	*Electronic prosthetic hand*	github.com/openbiomedical/OBM-FABLE
InMoov Hand	*Robotic hand build guide*	http://inmoov.fr/hand-and-forarm
Enabling the Future	*3D printed prosthetics*	http://enablingthefuture.org
OpenBCI	*Open source brain computer interface*	https://openbci.com
OpenAPS	*Automated pancreas system*	https://openaps.org
StarCat	*Open source bio signals*	https://starcat.io
Neuroon	*Open sleep/neuro tracker*	https://github.com/inteliclinic

SPACE

Ultrascope	*Automated robotics observatory*	http://openspaceagency.com
Tiny Radio Telescope	*DIY radio telescope*	https://hackaday.io/project/161556
Loop Antenna	*Small loop antenna guide*	https://opensourceradiotelescopes.org
Horn Antennas	*Two horn antenna guides*	https://opensourceradiotelescopes.org
SatNOGS	*Global network of satellite ground-stations*	https://satnogs.org
UPSat	*QB50 cubesat by the Libre Space Foundation*	https://upsat.gr

SPACE

Space Decentral	*Building a decentralized space program*	`https://spacedecentral.net`
Blockstream Satellite	*Satellite based bitcoin transactions*	`https://blockstream.com/satellite`

HOUSING

Wikihouse	*Open source house building designs*	`https://wikihouse.cc`
Open Building Inst.	*Open source furniture plans and more*	`https://openbuildinginstitute.org`
No Throw Design	*Downloadable furniture plans and instructions*	`https://nothrowdesign.com/you-make/`
Open Source Ecology	*Plans for building an entire village*	`https://opensourceecology.org/gvcs`
Divine on the Road	*Plans for building your dream van*	`https://divineontheroad.com/build-a-van`
The Vanual	*DIY Campervan Conversion*	`https://thevanual.com`
Obrary	*Furniture designs requiring CNC mill/laser*	`https://obrary.com`

COMMUNICATIONS

LibreCMC	*Embedded OS, supporting a range of routers*	`https://librecmc.org`
OpenWRT	*Open Source wireless router firmware*	`https://openwrt.org`
DAPNet	*Decentralized Amateur Paging Network*	`https://hampager.de`
OpenVPN	*VPN software for secure data communications*	`https://openvpn.net`
LimeSDR	*Open source software defined radio board*	`https://github.com/myriadrf`

COMMUNICATIONS

HackRF	*Low cost open source SDR platform*	https://github.com/mossmann/hackrf
Project Byzantium	*Linux-based emergency mesh networking*	http://project-byzantium.org
Doge Microsystems	*Build Your Own Dial Up ISP*	dogemicrosystems.ca/wiki/Dial_up_server
Low-Tech Magazine	*How to build a solar powered website*	https://solar.lowtechmagazine.com
Subnodes	*Turning Raspberry Pi's into access points*	http://subnodes.org
Disaster Radio	*Solar powered, disaster-resistant comms network*	https://disaster.radio
OpenWISP	*An open management system for wireless ISPs*	http://openwisp.org

MANUFACTURING / LASER CUTTER

LS-Laser	*Full guide for creating a CO2 laser cutter*	openbuilds.com/builds/ls-laser.7304
Lasersaur	*Build guide for Lasersaur cutter*	https://github.com/nortd/lasersaur/wiki
Laser V	*Smaller footprint laser engraver*	openbuilds.com/builds/much4-laserv-printed-version.937
LaseDuo	*Large, heavy duty laser cutter*	http://laserduo.com

MANUFACTURING / 3D PRINTING

Creality Ender-3	*Hardware and software plans for Ender-3 printer*	github.com/Creality3DPrinting/Ender-3
Creality CR-10	*Hardware and software plans for CR-10 printer*	github.com/Creality3DPrinting/CR-10
Falla 3D	*Open source 3D printer that uses maglev system*	https://github.com/3dita/Falla3D

MANUFACTURING / 3D PRINTING

Indie i2	*Small footprint 3D printer*	openbuilds.com/builds/indie-i2.1976
Infinite 3D Printer	*Design for an automatic, conveyor 3D printer*	https://hackaday.io/project/114738
C-Bot	*Core XY style 3D printer with large print bed*	openbuilds.com/builds/c-bot.1146
3Drag	*3D printer which uses the RepRap philosophy*	https://reprap.org/wiki/3drag

MANUFACTURING / CNC MILLS

C-beam Sphinx	*Large aluminium based CNC router*	openbuilds.com/builds/c-beam-sphinx.3605
OpenBuilds OX CNC	*Extremely detailed CNC build guide*	openbuilds.com/builds/openbuilds-ox-cnc-machine.341
OpenBuilds MiniMill	*Small footprint desktop CNC*	openbuilds.com/builds/openbuilds-minimill.5087
X-Carve	*Popular modular CNC designs*	x-carve-instructions.inventables.com
Maslow CNC	*Community driven large format CNC*	https://maslowcnc.com
AE1 CNC	*Small footprint engraver/CNC*	github.com/Chris-Annin/AE1-CNC-engraver-router
Shapeoko	*The original open source desktop CNC*	https://wiki.shapeoko.com

MANUFACTURING / 3D SCANNING

MakerScanner	*Open source 3D laser scanner*	http://makerscanner.com
Ciclop	*Printable 3D scanner project*	https://github.com/bqlabs/ciclop
FabScan	*Raspberry Pi based laser scanner*	http://fabscan.org

MANUFACTURING / VACUUM FORMING

Vacuum Former	*Open source vacuum former build guide*	https://labs.tcbl.eu/projects/32

ELECTRICITY

OSSI	*Open source solar inverter project*	https://github.com/transistorgrab/OSSI
Green Optimistic	*Various generator design guides*	greenoptimistic.com/category/green-tech-2/how-to
Wind Generator	*55 watt 3D printed generator design*	https://hackaday.io/project/87345
Portal Point Generator	*Compact 100+ watt generator*	https://hackaday.io/project/159568

FARMING

FarmBot	*Automated agriculture platform*	https://farm.bot
Food Computer	*Desktop food growing system*	http://openag.media.mit.edu/hardware
Farm Hack	*Various farming tool projects*	http://farmhack.org/tools

CRYPTOCURRENCY

Firefly	*Airgapped Ethereum hardware wallet*	https://firefly.city
RaspiBlitz	*Easy to use Lighting Node on RPi*	https://github.com/rootzoll/raspiblitz

DRONES

Flone	*Complete plans for building a drone*	`http://flone.cc/comprar-flone`
Paparazzi UAV	*Open source drone autopilot systems*	`http://paparazziuav.org`
ArduPilot	*Open source autopilot*	`http://ardupilot.org`
Dronecode	*Open source platform for UAVs*	`https://dronecode.org`

VIRTUAL/AUGMENTED REALITY

HardlightVR	*Open source haptic feedback suit*	`https://github.com/HardlightVR`
Relativity Headset	*VR Headset that costs ~$100*	`https://relativty.net`
Project North Star	*Leap Motion's AR headset*	`github.com/leapmotion/ProjectNorthStar`
OSVR	*The original open source VR headset*	`http://osvr.org`
Pupil	*Modular eyetracking platform*	`https://pupil-labs.com/pupil`
HoloKit	*Lost cost mixed reality platform*	`https://holokit.io`

COMPUTING

MNT Reform	*Open source, modular laptop computer*	`https://source.mntmn.com/MNT`

IN CLOSING: THE SATOSHI MINDSET

Satoshi Nakamoto is a symbol of hope. Not because the pseudonymous creator of Bitcoin birthed a brand new currency, a payment system, or even that he solved an ages old computer science problem, no it's much deeper than that.

On January 03 2009, block 0 of a new distributed system was mined. Contained within, was a message, *'The Times 03/Jan/2009 Chancellor on brink of second bailout for banks'*, revealing some of the thinking behind its genesis.

Taking into account that this was in the midst of the financial crisis, and combining that with later writings, we can infer that Satoshi wasn't happy with the economic system he found himself in.

What's interesting though, is that instead of moaning, begging, lobbying politicians, voting, protesting, occupying wall street, or any of that--ALL which I might add, never achieved any of their desired outcomes--he dared to create an alternative, and asked permission from no-one.

Even though we still live in this imperfect world, with many of the same corrupt forces, you could argue that Bitcoin has already succeeded in that it:

- Created a censorship-resistant system used by millions of people
- Created the most powerful computer hashing network in existence
- Created a market valued at multiple billions, all from nothing
- Has been used as a safe haven by citizens in failed states around the world
- Spawned a whole raft of other important software innovations

As Peter Thiel would say, that is a 0 to 1 development. Satoshi started all of this when he sat down and figured out how to solve a seemingly unsolvable problem.

He's quoted as once saying "I'm better with code than with words", and this is the core of

the mindset: creating new technological alternatives, instead of just complaining. Echoing this sentiment back in 1993 with the *Cypherpunk Manifesto*, Eric Hughes famously declared, "Cypherpunks write code."

This amalgam of recognizing problems, having technical skills to potentially fix them, and then taking personal ownership for trying to do so is probably one of, if not *the* most important qualities our species has.

Human history is littered with these feats--monumental forces of will which flip the script, change reality, and reveal new pathways for being.

Another example is Linux. Do you think the operating system would be used on the majority of phones, tablets, and servers today if not for Linus Torvalds deciding one day to create and release his own kernel, even when it seemed like a insane undertaking?

It's not about idolizing Satoshi or others. The point is that in order to solve big problems, we must first look inwards and take responsibility. Think: "What can I create to solve this?", and "What can I build that will be so good that others will voluntarily abandon their old ways for this new thing?"

Our civilization's heavy reliance on electronic technology means that now more than ever, you have a chance to come up with real technological solutions and nudge our species in a better direction.

So in closing, I just want to get across that it is down to **YOU**, and no one else. Stop shaking your fist at the sky, and shift your mindset.

Satoshi took on the responsibility of an impossible task, and succeeded. You can too. Pick a difficult problem, and go after it. If you need help, find like-minded people, and if you have gaps in your knowledge, learn.

This is how we improve things--everything else is a distraction.

I just wanted to say THANK YOU for reading NODE Vol 01, hope you liked it. Now it's your turn to go out there and build something.

NODE

SUBMISSIONS

We're going to open up submissions to the community for the next issue, so if you are working on something unique, new, and/or generally interesting, get in touch.

GUIDELINES:

- Original, unpublished content only
- Priority is on do-it-yourself, engineering, and those who are building interesting hardware and/or software
- No politics
- No illegal material
- We reserve the right to say no

If you're interested, email Mike: `editor@n-o-d-e.net`

See you next issue!